ARMY RESEARCH LABORATORY

Agent-Based Modeling of a Network-Centric Battle Team Operating Within an Information Operations Environment

by Brian G. Ruth and J. Dana Eckart

ARL-TR-2913 **February 2003**

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5068

ARL-TR-2913 **February 2003**

Agent-Based Modeling of a Network-Centric Battle Team Operating Within an Information Operations Environment

Brian G. Ruth
Survivability/Lethality Analysis Directorate, ARL

J. Dana Eckart
Virginia Bioinformatics Institute

Acknowledgments

The authors would like to thank Mr. Richard Sandmeyer of the U.S. Army Research Laboratory for his ongoing technical consultation during the course of the research presented in this report and for his careful and insightful technical review of the report.

Contents

List of Figures

List of Tables

Executive Summary

In this report, a model developed to analyze the emergent operational impact of hostile information operations (IO) stress events directed at a network-centric battle team is presented. In such a battle team, there is an operational triad of military roles that combat platforms can fulfill; they are as follows:

- sensors that sample the battlespace to collect data on friendly platforms, enemy platforms, and environmental conditions,
- commanders that assess situational data collected by sensors in order to decide upon responsive military actions, and
- actors that execute the military actions dictated by commanders.

This triad forms the basis of a distributed "system-of-systems" of battlefield entities connected via information flows, where each type of entity occupies a specific operational "niche." It should be noted that the network-centric battle team is characterized by its extensive use of networks to share information and allow for a coherent picture of the battlefield to emerge within all networked battlefield entities.

In order to simulate the operational dynamics of a military network-centric system-of-systems, the concept of agent-based models is introduced. In general, an agent is an autonomous computational entity with a perpetuated internal state and associated set of rules governing behavior. The agent's state is usually represented as a dynamic vector describing metrics such as agent position, identity, current functionality, and so on. A colony of agents can interact with one another by passing messages between themselves, which can represent communication, cooperative actions, or conflict. Given these elements, an agent-based model is then a collection of interacting agents contained within a virtual "artificial world." J. H. Holland[1] describes seven basic attributes of complex adaptive systems that should be reflected in a multiagent model of a network-centric system-of-systems; they are aggregation, nonlinearity, information flows, diversity, interagent tagging, access to internal models, and the use of building blocks. Taken together, these attributes allow for the emergence of a combat unit of distributed agents (platforms) that individually boasts specific functional capabilities that can be collectively exploited by the unit via networked communications.

An agent-based model of a network-centric system-of-systems can be implemented by incorporating local-rule-based agent dynamics patterned after cellular automata (CA) models. In general, a CA consists of a possibly infinite, *n*-dimensional regular lattice of cells. Each cell can

[1] Holland, J. H. *Hidden Order: How Adaptation Builds Complexity*. Reading, MA: Perseus Books, pp. 10–37, 1995.

be in a measurable state chosen from a finite alphabet (i.e., a set of possible cell states). The states of all cells in the lattice are updated simultaneously in discrete time steps using a transition function, which takes as its input the current state of the center cell and some finite collection of nearby cells that lie within a finite distance, collectively known as a "neighborhood." Cell neighborhoods may be either fixed or variable over time (but fixed within a time step); these neighborhoods are usually defined to be local to a center reference cell, but can extend beyond this conventional limit if necessary. Within this context, agents can then be defined as mobile automata (with perpetuated internal states and autonomous/semiautonomous operational behaviors) that move throughout a CA lattice and communicate with each other via message passing. Note that this combination of cell neighborhoods and agents can effectively model an operational environment and the entities that inhabit that environment, respectively.

The optimal configuration of individual agents within a combat unit (network-centric or otherwise) can be explored through the use of genetic algorithms (GA), a class of heuristic search methods that mimic and exploit the genetic dynamics underlying natural evolution to search for optimal solutions of general combinatorial optimization problems. Given an optimization problem, GAs provide a mechanism for efficiently exploring the problem solution space for "good solutions" based on one or more "goodness of fit" criteria called fitness functions. Possible solutions to the optimization problem of interest are encoded as chromosomes (or sometimes as a set of correlated chromosomes called a genome), where a chromosome is a string of problem-relevant variables or genes. The GA then evolves one population of chromosomes (or genomes) into another according to their fitness by employing an iterative "evolutionary" process. For the current application, the GA will need to coevolve the configurations of adversarial units of Blue and Red agents engaged in combat by simultaneously optimizing hardware-based capability and behavior assignment to individual agents in both units based on combat performance.

The first step in building a CA-based combat agent unit or battle team is to define the attributes of a generic combat agent within the battle team. Within the context of a CA lattice, a combat agent is an autonomous or semiautonomous automaton with a dynamically-perpetuated internal functional state vector representing agent mobility, firepower, sensor, and communication capabilities, as well as a set of behavioral rules defining how an agent interacts with other combat agents operating within a shared two-dimensional (2-D) regular lattice. The combat agent state vector maintains and updates the following information fields:

- mobility capability status,
- firepower capability status,
- on-board sensor functional parameters,
- supply status of on-board unattended ground sensor (UGS) units, radio frequency (RF) battery-powered "jammer bombs," and land mines,

- an internal sensor message database representing the combat agent's situational awareness (SA) of the virtual battlespace,
- interagent communication capability status (transmission and reception functionality), and
- rules of behavior that define the degree of autonomy within the combat agent.

Combat agent movement is then modeled as the mapping of this agent state vector into adjacent CA lattice cells during a subsequent simulation time cycle (i.e., a periodic sequence of events that occurs over a contiguous set of simulation time steps).

In addition to a dynamic state vector, each combat agent also maintains a set of functional ranges associated with the following capabilities:

- the maximum distance an on-board sensor can acquire information on neighboring combat agents,
- the maximum distance a transmitting agent can communicate with receiving neighboring agents,
- the maximum distance an agent with firepower capability can engage a neighboring agent,
- the maximum distance a jammer-equipped agent can jam the communication reception capability of neighboring agents, and
- the maximum distance an agent can move in one simulation time cycle.

For purposes of simplification, these ranges define a set of concentric extended Moore neighborhoods relative to the 2-D CA lattice, where the reference combat agent occupies the center cell within the neighborhood.

The next step in building the combat agent battle team is to design a framework with which to represent the capabilities and behaviors of the entire team. Invoking genetic algorithm terminology, the battle team genome represents the capability/behavioral configuration of the combat agent collective unit. Each genome is constructed of a set of chromosomes, where a chromosome represents the capability/behavioral configuration of a single combat agent. Each combat agent chromosome, in turn, is constructed from a set of genes representing individual agent capability and behavioral attributes. Finally, each gene is characterized by a finite set of alleles or permissible gene states. Note that although each combat agent chromosome within a battle team genome is constructed by using the same schema or gene template, the allele ranges for specific genes can vary from chromosome to chromosome. Once the genes within a generic combat agent schema have been specified, the rules governing how these genes are utilized by combat agents within a dynamic CA simulation can be delineated.

Once the CA-based combat agent simulation engine has been designed, the development and demonstration of a software framework that utilizes the CA engine for IO-stressed Blue/Red

combat simulations can be discussed. Due to the prolonged runtimes necessary to simulate densely packed UGS networks and multihop message relaying within the CA implementation of combat agent communication nets, the decision was made to codevelop two different versions of the IO simulation software framework.

In the first simulation framework, combat-motivated coevolution of both Blue and Red unit configuration genomes is simulated within an IO-stressed virtual environment. In this framework, the duration of an update time cycle within the CA simulation engine is compressed to facilitate the thousands of Blue/Red combat instances necessary to execute the genomic coevolution. Initial populations of Blue/Red candidate genomes (or candidate combat unit configurations) are randomly generated and then filtered using hardware cost constraints. These candidate genomes are then input into the `combat_ga` CA simulation engine along with a virtual landscape and a territorial importance gradient (that serves to motivate combat agent movement towards a goal), and a combat simulation is executed. Such combat simulations are subsequently run for each possible Blue/Red genome pairing (or a subset thereof).

During a Blue/Red combat simulation, a data filter dynamically calculates the dynamic operational "fitness" (analogous to one or more combined combat unit measures of performance averaged over a combat mission time interval) of each opposing combat agent unit within the simulation. These combat simulations can include hostile IO actions (which, in this case, are limited to RF jamming by jammer-equipped combat agents). The output of the data filter is then either directed into a GA or can be directly analyzed in time series format. In the former case, the top five "fittest" genomes for each side are determined (based on the collective results from all instantiated Blue/Red combat simulations) and then used by the GA to create new "offspring" genome populations (through the use of algorithmic "genetic operations" performed on genome pairs working in combination with a cost filter). These replenished populations are used to continue the coevolutionary process across successive genome "generations"; this process is iteratively repeated until both Blue and Red genome fitness values converge at or near a maximal value. The genome populations can also be dynamically classified (as a function of hardware-based capabilities and behaviors encoded into a genome) throughout a coevolutionary simulation run.

Genomic combat fitness is the measure of performance used to guide the Blue/Red coevolutionary process. The fitness of a combat unit is an operational measure of its collective ability to successfully execute combat actions and is analogous to one or more combined combat unit measures of performance. Fitness is thus a metric that can be used to evaluate the operational effectiveness of a specific genomic configuration vector encoding of a combat unit's hardware capabilities and behaviors. To carry this out, a set of genome fitness functions or dynamic measures of performance was designed. These functions are evaluated at the end of each simulation time cycle update of the CA engine and represent the collective performance of

all agents within a combat unit. The linear superposition of all fitness function metrics used to measure the perpetuation of combat unit collective capabilities throughout a simulation is called the survival fitness, which provides an objective function that can be optimized by the GA.

A coevolutionary Blue vs. Red simulation was run on a 32-CPU SGI Origin 3800 computer, with Blue and Red units programmed to execute defender and invader combat roles, respectively. Here, Blue/Red combat instances for each evolutionary generation of candidate genomes were run in parallel, after which calculated survival fitness values were centrally averaged and reported. Three combat simulation instances were executed (and then averaged) for each Blue/Red genome candidate pairing, with 30 pairings per generation. The latter situation was iteratively realized by randomly selecting one candidate genome from both Blue and Red genome pools, for a combat pairing, removing the selected genomes from their respective pools and repeating the random selection process until all genomes have been paired for combat. Each combat simulation was run for 300 time cycles, at which point values of the time-averaged survival fitness for both Blue and Red genomes were evaluated and reported. Then, for both Blue and Red units, the top five candidate genome scorers from each generation were used to produce the next generation of 30 candidate genomes via random mating pairs.

Coevolutionary simulation results were collected across 1360 successive generations of Blue and Red combat unit genomes. In these results, Red quickly "discovers" unit configurations that serve to maintain its average fitness within the interval [0.38, 0.43], which, in turn, motivates Blue to compensate with new configurations that likewise drive average Blue fitness to the interval [0.49, 0.59]. This coevolutionary process nicely illustrates the "Red Queen Principle" from evolutionary biology, where for an evolutionary system, continuing development is needed just in order to maintain its fitness level relative to the systems it is coevolving with. Since, due to computational runtime constraints, the coevolutionary process was prematurely terminated before any strong evidence of survival fitness convergence was evidenced, it remains unclear whether the Blue and Red combat units would ever break out of the "Red Queen" oscillatory pattern illustrated in the figure. If the current theory of coevolutionary neutral mutations (where gene mutations produce insignificant variations in relative fitness) is applicable within the current context, then fitness convergence might never be achievable without relaxing some of the gene allele and cost constraints defined for this demonstration simulation.

Evidence of continuing development of both Blue and Red combat units throughout the demonstration simulation is provided by a classification analysis of Blue and Red genome populations. This type of analysis involves first setting up sets of combat agent types or "classes" specified by agent chromosome gene values and then observing the correlated coevolutionary dynamics of optimal Blue and Red unit configurations relative to these sets of classes. Analysis results using a first set of agent classes (based on different combinations of sensor/firepower ranges) show that the Blue unit configuration is observed to oscillate between variable mixtures of medium- and long-range sensor agents with variable-range firepower weapon capability, finally settling into a mode where most agents have medium-range sensor and

short-range firepower capability. The Red unit, on the other hand, which must execute the more operationally challenging primary role of invader, coherently explores sensor/firepower classes until finally settling on a medium-range sensor and short-range fire for most agents.

Next, using a second set of combat agent classes specified by agent behavioral weighting combinations, Blue is seen to haphazardly explore various classes, then settle on a passive/indecisive behavioral mode for over 600 generations, and finally switch over to a mixture of passive/indecisive and aggressive/ indecisive agents at the end of the coevolutionary run. This is not surprising, in that Blue's primary mission is defensive, and thus Blue agents can afford to be "indecisive" (effectively immobile) and less "aggressive" than Red agents. After an initial exploratory period, Red, on the other hand, first "discovers" a behavioral configuration where all agents act in an aggressive (firing a weapon without provocation) and noncovert (moving directly towards Blue agents) manner that serves to maintain its operational fitness. Then, when Blue counters with behaviors that stress Red fitness, Red, in turn, progressively adopts passive and noncovert, aggressive and noncovert, aggressive but indecisive (trying to move both towards/away from Blue), and aggressive but covert modes of behavior, finally settling on the last mode (with a small fraction of the Red unit continuing to explore other behaviors). Again, Red faces the greater evolutionary pressure since its primary mission is invasive in nature and thus benefits by coherently moving towards better strategic behavioral modes.

In the second IO simulation software framework, a quasi-scripted (and thus nonevolving) combat agent C2 process is directly embedded into the `combat_proto` CA simulation engine. This second and enhanced version of the CA simulation engine models the deployment and function of notional internetted UGS units and multihop message relaying and thus extends the network-centric capabilities of `combat_ga`. In this new framework, hand-scripted configuration genomes were constructed in order to represent a Blue unit of networked mobile platforms making up a network-centric combat unit as well as a Red unit of loosely organized nonnetworked platforms. Combat simulations were run using the `combat_proto` CA engine in combination with routines to generate a simulation landscape (based on the notional "Fombler's Ford" terrain designed for the Future Combat Systems concept illustration), a territorial importance gradient superimposed over the landscape, and to initially position Blue and Red combat agents. Then, a data filter dynamically calculated preselected fitness functions of each opposing combat agent unit within the simulation, which can include hostile IO actions. Finally, the fitness function outputs of the filter were analyzed in a time series format (as opposed to weighting the functions and then combining them into the survival fitness objective function as was done in the previous framework).

A sensitivity analysis was conducted using the `combat_proto` CA engine in conjunction with the "Fombler's Ford" scenario in order to explore the sensitivity of the following model parameters to IO stress:

- UGS density - the average packing density of the peripheral Blue UGS network.
- Blue fire point - the outward distance from the perimeter of Blue territory at which point Blue combat agents are allowed to fire on acquired Red targets.
- Blue shots/cycle - the number of shots a firepower-equipped Blue combat agent can fire during one simulation time cycle.

This sensitivity analysis was predicated on the time series response generated by the following four particular fitness functions as applied to specific combat units:

- Blue and Red combat unit survival levels,
- Blue's ability to sense the battlefield,
- Blue's ability to prevent Red penetration into Blue territory, and
- collective Blue situational awareness of Red.

In this set of simulations, these fitness functions serve as dynamic measures of collective combat unit performance.

Time series results from six different "Fombler's Ford" scenario variants (run in both benign and IO-stressed operational environments) were generated in the sensitivity analysis. In these scenario variants, the analyzed model parameters were set to values within the following ranges: (1) UGS density = 0.55 – 1.00, (2) Blue fire point = 10 – 17 cells, and (3) Blue shots/cycle = 1 – 3. Multiple simulation instances of each scenario variant were run using the set of 20 sequential random seeds {0, 1, 2, ..., 19}, and then time series results were averaged and plotted. The time series results clearly illustrated a (not surprising) monotonically increasing combat effectiveness and survivability of the Blue unit as the model parameters were increased from minimal to maximal values. The results also clearly illustrated that the amount of operational degradation experienced the Blue unit in the presence of Red jammer bombs was increasingly attenuated as a function of increasing model parameter values. Thus, the sensitivity analysis demonstrated how providing the Blue unit with a combination of enhanced operational capabilities could act to significantly attenuate the perturbative impact of a form of IO stress that results in denial of communication network connectivity.

Given that the combat agents used within the CA models are reactive in nature (with limited decision-making capability and no explicit "commander" type of agent able to formulate new tactical strategies when operationally required), the next step in model development is to provide agents with more deliberative behaviors, thus providing a framework for the hierarchical C2 decision-related structures ubiquitous to all military combat units. Current research into building more complex varieties of decision-making military command agents as well as hostile terrorist-

style agents operating within densely populated urban areas can provide a foundation for designing deliberative combat agents. Further work is still required, however, in order to realize rapidly adaptive agents operating within a network-centric combat structure.

1. Introduction

1.1 Purpose

The cellular-automata-based combat agent model presented in this report was developed in order to analyze the emergent behavior exhibited by a network-centric battle team exposed to hostile information operations (IO) stress in the form of radio frequency (RF) communications jamming.

1.2 Background

1.2.1 Network-Centric Complex Adaptive Systems

A complex system can be defined as a group of many interacting parts functioning as a unified whole. Such a system is distinguishable from its environment by recognizable boundaries, where the global behavior of the whole system is generally not equal to the sum of the parts. Even if the individual behavior of an isolated part is linear, the collective behavior that emerges globally through networked interactions and interdependencies among the parts is often nonlinear. This collective behavior arises from the detailed structure, behavior, and interpart relationships as they are defined on a finer scale, all of which dynamically evolve within the context of an irreducible whole. Furthermore, a complex adaptive system can then be defined as a complex system whose interacting parts can adapt to changing operational environments.

The paradigm of complex adaptive systems is directly applicable to a relatively new concept in military operations enabled by advances in information technologies known as network-centric warfare (NCW). Perhaps the best way to define the concept of NCW is through listing its basic tenets (Herman and Hayes 1999) as follows:

- Higher military echelons provide objectives, timelines, intent, and resource planning.
- Bottom-up self-synchronizing execution allows all weapons and sensors to be available to all subscribers on the same or linked networks.
- A high level of shared situational awareness enables execution decisions to be coordinated without significant upper echelon intervention.

These tenets suggest the following operational triad of military entities (Alberts et al. 2000):

- sensors that sample the battlespace to collect data on friendly platforms, enemy platforms, and environmental conditions,
- commanders that assess situational data collected by sensors in order to decide upon responsive military actions, and
- actors that execute the military actions dictated by commanders.

This triad is illustrated in Figure 1, which depicts a distributed "system-of-systems" of battlefield entities connected via information flows, where each type of entity occupies a specific operational "niche." It should be noted that NCW is not characterized as warfare by networks, or against networks but rather as the extensive use of networks to share information and allow for a coherent picture of the battlefield to emerge within all networked battlefield entities.

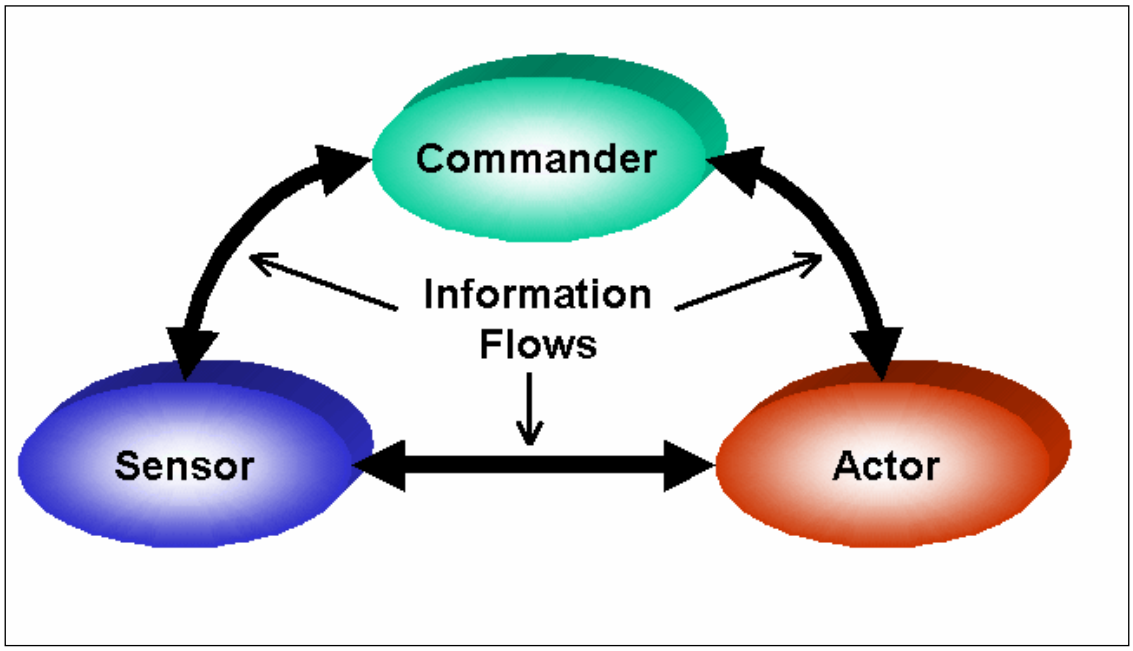


Figure 1. Triad of distributed battlefield entity types within a network-centric "system-of-systems."

To analyze a network-centric "system-of-systems" as a complex adaptive system operating within an equally-complex IO environment, the following three distinct scales define the structure, behavior, and relationships: (1) the microscale, which addresses interactions and interdependencies between information system (IS) components (e.g., 1553B data bus, combat net radios, and personal computers) within a platform, (2) the mesoscale, which addresses local interactions/interdependencies between networked battlefield platforms, and (3) the macroscale, which addresses the holistic global behavior of the collective system-of-systems structure (zum Brunnen et al. 2000). Usually, IO threat/target system interactions occur at the micro- and/or mesoscale levels, affecting information flows between IS components within a platform and/or flows between locally networked platforms, respectively. In the current model, the emergent global or macroscale impact of IO stress, which is directed at the mesoscale networked platforms making up a combat unit or "battle team," is analyzed.

1.2.2 Agent-Based Models

In order to simulate the operational dynamics of a military network-centric complex adaptive system, the concept of agent-based models is introduced next. In general, an agent (see Figure 2) is an autonomous computational entity with a perpetuated internal state and associated set of rules governing behavior (Maes 1990). The agent's state is usually represented as a dynamic vector describing metrics such as agent position, identity, current functionality, and so on. A colony of agents can interact with one another by passing messages between themselves, which can represent communication, cooperative actions, or conflict. Given these elements, an agent-based model is then a collection of interacting agents contained within a virtual "artificial world" (see Figure 3). Agent interactive behavior is dictated by sets of if/then "production rules," which are often specific to different agent types and define strategies for dealing with other agents as well as a possibly stressful operating environment. Also, agents are often organized into some type of hierarchical organization. Finally, the time-series output of an agent-based model can be viewed directly via a graphical user interface (GUI) or processed through a data filter.

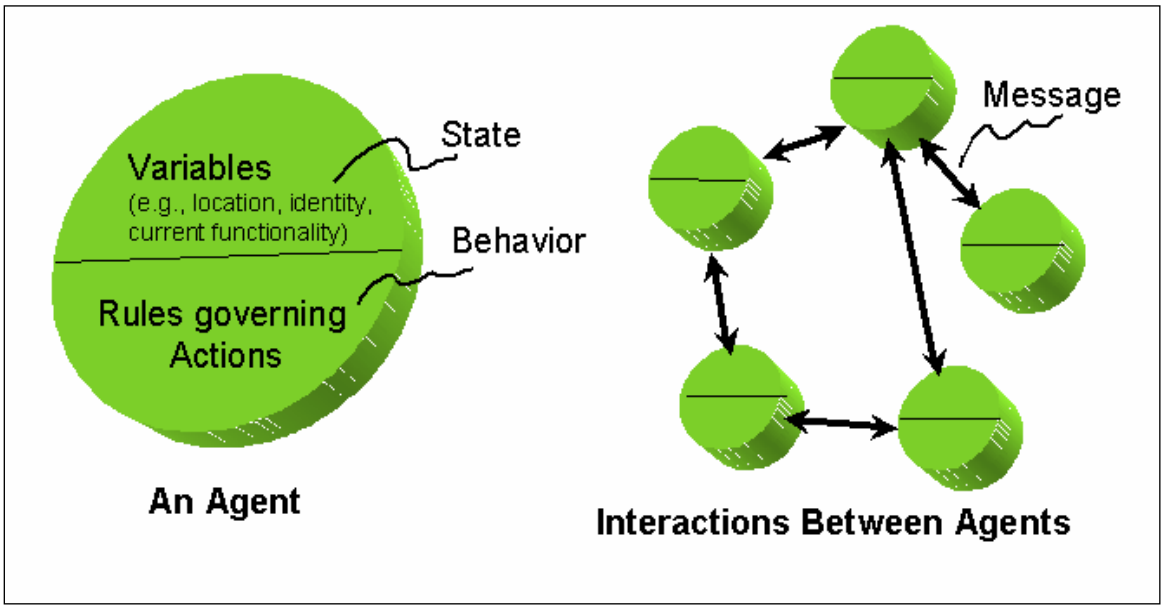


Figure 2. Representation of a generic agent.

Holland (1995) describes the following seven basic attributes common to all complex adaptive systems that thus should be reflected in any multiagent model of such systems:

(1) aggregation, which involves the categorization of information at the agent level and emergent behavior at the macroscopic multiagent level,

(2) nonlinearity, where the whole is often more or less than the sum of the parts,

(3) information flows between interacting agents,

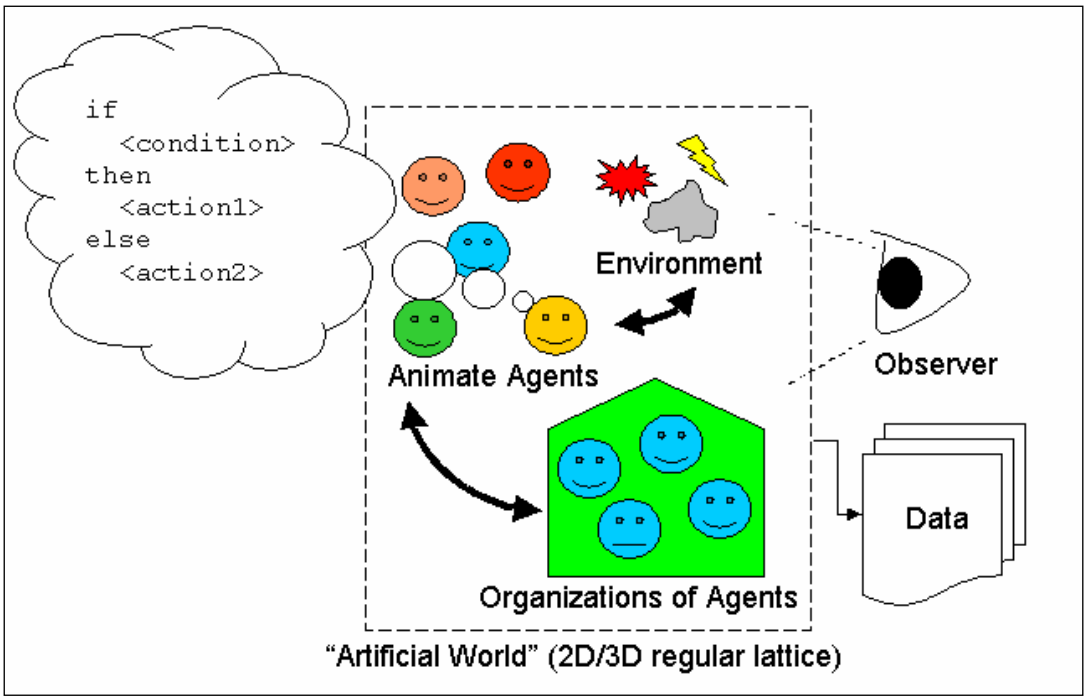


Figure 3. Representation of a generic agent-based model.

(4) diversity among agents (i.e., multiagent systems are heterogeneous),

(5) tagging to identify individuality among agents,

(6) internal models that specify how agents react to other agents and/or the operational environment, and

(7) building blocks, which are the components or primitives used to construct agents.

Conceptually, an agent-based model of a network-centric system-of-systems would need to demonstrate each of these attributes, where distributed agents (platforms) individually boast specific functional capabilities that can be collectively exploited by the combat unit via networked communications. Existing agent-based models of land warfare such as Irreducible Semiautonomous Adaptive Combat (ISAAC) (Ilachinski 2000), Simulation of Information in Battlefield Decisions (SinBaD) (Hencke 1998), and AgentKit (Woodaman 2000) all address the emergent behavior of combat units of interacting Blue and Red agents, but neglect the aforementioned network-centric "sensor/commander/actor" operational triad. What is still needed is a model where agents collect and fuse situational awareness data from distributed sensory devices and then execute subsequent combat actions based on these data.

On a design level, an agent-based model of a network-centric spatially-dispersed combat unit would need to address both the semiautonomous dynamics of individual agents as well as the adaptive behavior of such agents (which is driven by the necessity for an agent to adapt to the adaptation patterns of other agents). This model structure can be implemented by incorporating the following:

- local-rule-based agent dynamics patterned after cellular automata models and
- parameterization of the local decision space and the formulation of strategy and/or tactics patterned after genetic algorithm models.

These two types of models will be discussed in the next subsections.

1.2.3 Cellular Automata

In general, a cellular automata (CA) (von Neumann 1966) consists of a possibly infinite, *n*-dimensional regular lattice of cells. Each cell can be in a measurable state chosen from a finite alphabet (i.e., a set of possible cell states). The states of all cells in the lattice are updated simultaneously in discrete time steps using a transition function, which takes as its input the current state of the center cell and some finite collection of nearby cells that lie within a finite distance, collectively known as a "neighborhood." Figure 4 displays two well-known CA cell neighborhoods in two dimensions, where C = center cell and N = neighbor cell. Cell neighborhoods may be either fixed or variable over time (but fixed within a time step); these neighborhoods are usually defined to be local to a center reference cell but can extend beyond this conventional limit if necessary. Within this context, agents can then be defined as mobile automata (with perpetuated internal states and autonomous/semiautonomous operational behaviors) that move throughout a CA lattice and communicate with each other via message passing. Note that this combination of cell neighborhoods and agents can effectively model an operational environment and the entities that inhabit that environment, respectively.

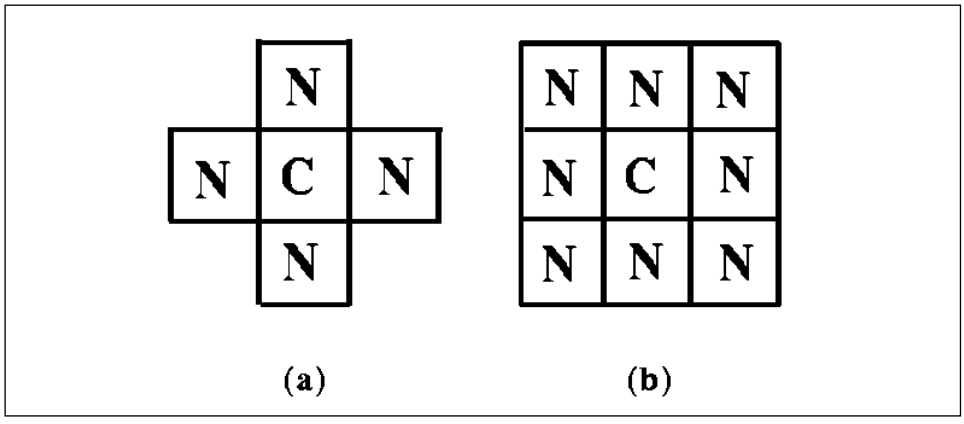


Figure 4. CA cell neighborhoods in two dimensions: (a) von Neumann neighborhood and (b) Moore neighborhood.

1.2.4 Genetic Algorithms

Genetic algorithms (GA) are a class of heuristic search methods that mimic and exploit the genetic dynamics underlying natural evolution to search for optimal solutions of general combinatorial optimization problems (Mitchell 1993). Given an optimization problem (e.g., maximizing a function over a specific interval), GAs provide a mechanism for efficiently exploring the problem solution space for "good solutions" based on one or more "goodness of

fit" criteria called fitness functions. Possible solutions to the optimization problem of interest are encoded as chromosomes (or sometimes as a set of correlated chromosomes called a genome), where a chromosome is a string of problem-relevant variables or genes. The GA then evolves one population of chromosomes into another according to their fitness by employing an iterative "evolutionary" process (Figure 5). This process involves the following iterative steps (Ilachinski 1996):

- Step 1: Begin with a randomly generated population of chromosome- (or genome-) encoded "solutions" to a given combinatorial optimization problem.
- Step 2*:* Calculate the fitness of each chromosome/genome, where fitness is a measure of how well a member of the population performs at solving the problem.
- Step 3*:* Retain only the fittest members of the chromosome/genome population and discard the remaining members.
- Step 4: Generate a new population of chromosomes/genomes from the remaining members of the old population by applying the genetic operations reproduction (the generation of a "child" chromosome/genome from two high-fitness "parent" chromosomes/genome), crossover (cutting two parent chromosomes into two subchromosomes per parent or cutting two parent genomes into two sets of chromosomes or subgenomes per parent and then mixing and reconnecting the subchromosomes or subgenomes to create two new children chromosomes or genomes, respectively) and mutation (randomly changing the gene values within a chromosome).
- Step 5*:* Calculate the fitness of these new members of the chromosome/genome population, retain the fittest members, discard the remaining members, and reiterate the process until a "maximally fit" solution is achieved.

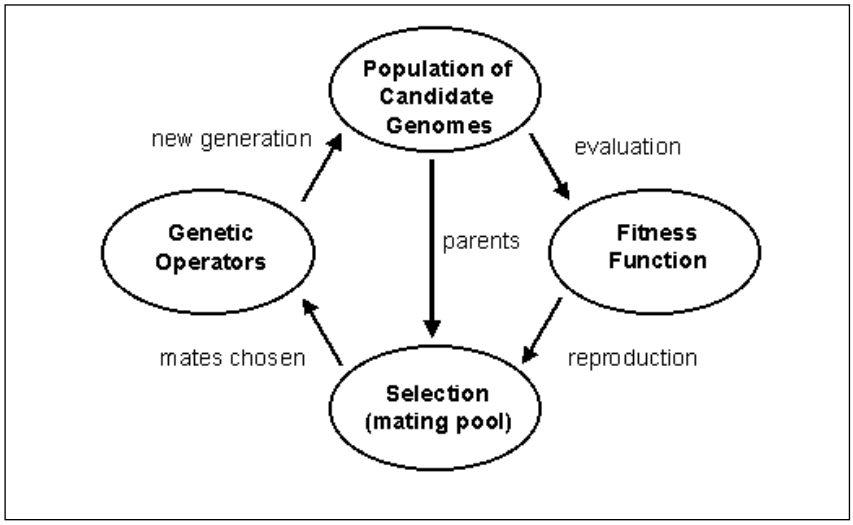


Figure 5. Iterative evolutionary processes within a genetic algorithm (from Ilachinski 1996).

This evolutionary process is often referred to as navigating through a fitness landscape, which is an $n + 1$-dimensional hypersurface that measures the fitness of all possible n-dimensional chromosome/genome configurations. Thus, convergence of a genetic algorithm towards an optimal chromosome/genome configuration is analogous to moving towards a maximal fitness point on the fitness landscape.

There have been several previous research efforts to apply GA technology to military optimization problems. Of these, the most relevant relative to an agent-based modeling application is the FOX & HARE model, which addresses the optimization of behavioral course-of-action (COA) selection within two opposed and coevolving Blue and Red military units (Hillis and Winkler 2000; Funes and Pollack 2000). In a FOX & HARE simulation, Blue and Red unit COAs were generated by genetic algorithms, with relative COA fitness evaluated via unit performance in a simplified wargame model. Then, new Blue and Red unit COAs were generated and played against each other, thus allowing new Blue/Red solutions to coevolve during each generational iteration. However, although it will address coevolving Blue/Red military units of agents, the model addressed in this report will focus on optimization of hardware-based capability and behavior assignment to individual agents rather than COA optimization as handled by a unit commander.

1.2.5 Threat

The threats addressed within this report are hostile IO stress events (U.S. Department of the Army 1996) that result in denial of communication network service within a network-centric battle team.

1.3 Scope

The applicability of the CA agent-based model presented in this report is constrained by the following assumptions:

- Networked platform nodes within a network-centric battle team treat all acquired situational awareness (SA) data as true and valid (i.e., combat actions based on the relative validity of data are not modeled).
- Human-based command and control (C2) of combat agents is very limited and thus is restricted to the criterion that “unmanned” combat agents may fire upon acquired enemy targets provided that the SA target data they use are provided to them by “manned” combat agents.
- IO stress will only impact a node’s ability to receive SA messages from other neighboring nodes and not affect internal information flows within a platform node.
- All combat agents are modeled as reactive in nature, where each agent decision process maps to one, and only one, possible output.

2. The Cellular Automata-Based Combat Agent Model

2.1 The Cellular Simulation System

The CA-based combat agent model is implemented through the use of the Cellular simulation system, an easy-to-use toolkit for the modeling of physical systems with cellular automata. The Cellular system consists of several modules as follows:

- a programming language, Cellang, and associated compiler, `cellc,`
- an abstract virtual cellular automata machine (`avcam`) for Cellang code execution, and
- `cellview`, a program for viewing simulation results in either two-dimensional (2-D) sections or in three dimensions with variable perspective.

Compiled Cellang programs can be run with input provided at any specified time during the execution. The results of an execution can either be viewed graphically as an output stream of cell locations and values or passed through a custom filter before being reported. The output stream, or a similar stream produced by a custom filter, can also be directed into `cellview` for viewing or may be passed through other programs that compile statistics, massage the data, or merely act as a valve to control the flow of data from the cellular automata program to the viewer.

Programs written in Cellang have two main components—a cell description and a set of statements. The cell description determines how many dimensions there are, the vector of fields associated with each cell, and the range of values that can be associated with each field. In addition to cell-specific fields, the cell description also specifies the various types and attribute fields of mobile agents that can reside in the cells. The set of statements, on the other hand, encode the transition rules for all lattice cells (which can be applied homogeneously to all cells or heterogeneously according to cell type or the type of agent[s] currently residing in the cell). For more detailed information on the Cellular toolkit, see Eckart (1992a, 1992b).

2.2 Building the Combat Agent Battle Team

2.2.1 The Generic Combat Agent

The first step in building a CA-based combat agent unit or battle team is to define the attributes of a generic combat agent within the battle team. Within the context of a CA lattice, a combat agent is an autonomous or semiautonomous automaton with a dynamically perpetuated internal functional state vector representing agent mobility, firepower, sensor, and communication capabilities, as well as a set of behavioral rules defining how an agent interacts with other combat agents operating within a shared 2-D regular lattice. A representation of a generic

combat agent is depicted in Figure 6. As illustrated in this depiction, a combat agent state vector maintains and updates the following information fields:

- mobility capability status,
- firepower capability status,
- on-board sensor functional parameters,
- supply status of on-board unattended ground sensor (UGS) units, RF battery-powered "jammer bombs," and land mines,
- an internal sensor message database representing the combat agent's SA of the virtual battlespace,
- interagent communication capability status (transmission and reception functionality, and
- rules of behavior that define the degree of autonomy within the combat agent.

Combat agent movement is then modeled as the mapping of this agent state vector into adjacent CA lattice cells during a subsequent simulation time cycle (i.e., a periodic sequence of events that occurs over a contiguous set of simulation time steps).

In addition to a dynamic state vector, each combat agent also maintains a set of functional ranges associated with the following capabilities:

- the maximum distance an on-board sensor can acquire information on neighboring combat agents,
- the maximum distance a transmitting agent can communicate with receiving neighbor agents,
- the maximum distance an agent with firepower capability can engage a neighbor agent,
- the maximum distance a jammer-equipped agent can jam the communication reception capability of neighboring agents, and
- the maximum distance an agent can move in one simulation time cycle.

For purposes of simplification, these ranges define a set of concentric extended Moore neighborhoods relative to the 2-D CA lattice (see Figure 4b), where the reference combat agent occupies the center cell within the neighborhood. Also, the range hierarchy depicted in Figure 6 is but one realization of permissible configurations (in other words, the ranges are flexible relative to one another).

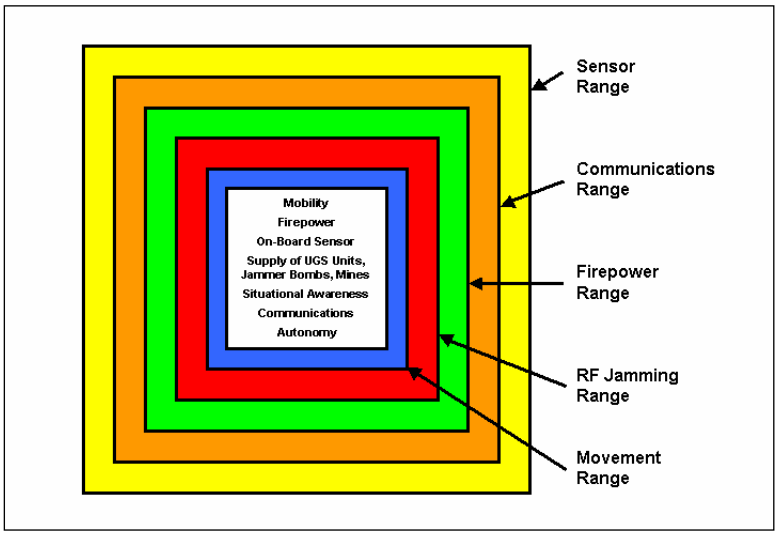


Figure 6. A generic combat agent.

2.2.2 Combat Agent Battle Team Representation

The next step in building the combat agent battle team is to design a framework with which to represent the capabilities and behaviors of the entire team. Figure 7 illustrates this representation framework. Invoking the genetic algorithm terminology first introduced in section 1.2.4, the battle team genome represents the capability/behavioral configuration of the combat agent collective unit. Each genome is constructed of a set of chromosomes, where a chromosome represents the capability/behavioral configuration of a single combat agent. Each combat agent chromosome, in turn, is constructed from a set of genes representing individual agent capability and behavioral attributes. Finally, each gene is characterized by a finite set of alleles or permissible gene states. Note that, although each combat agent chromosome within a battle team genome is constructed by using the same schema or gene template, the allele ranges for specific genes can vary from chromosome to chromosome.

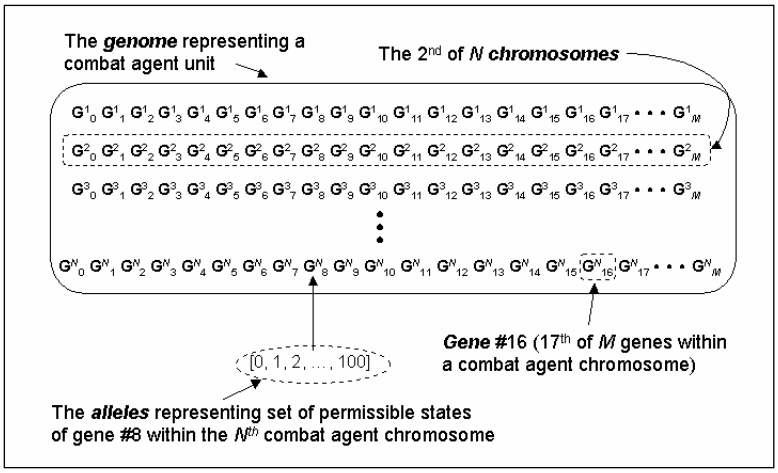


Figure 7. Representation framework for a combat agent battle team configuration.

Given a viable battle team framework within which to encode agent hardware capabilities and behaviors, the specific genes within a combat agent chromosome can now be assigned. The genetic schema template for encoding combat agent capability and behavioral states is shown in Figure 8. Within this schema, there are hardware-based capability classes addressing combat agent firepower, passive self-protection, on-board sensor, communications, on-board RF jammer, carried supplies of UGS units, mines, and RF jammer bombs, and mobility. Also included is a characterization of the combat agent's acquisition signature, as well as a behavioral "personality vector" encoding the battlefield tendencies of the agent to attack, retaliate, and move towards/away from friends/enemies and or friendly/enemy territory.

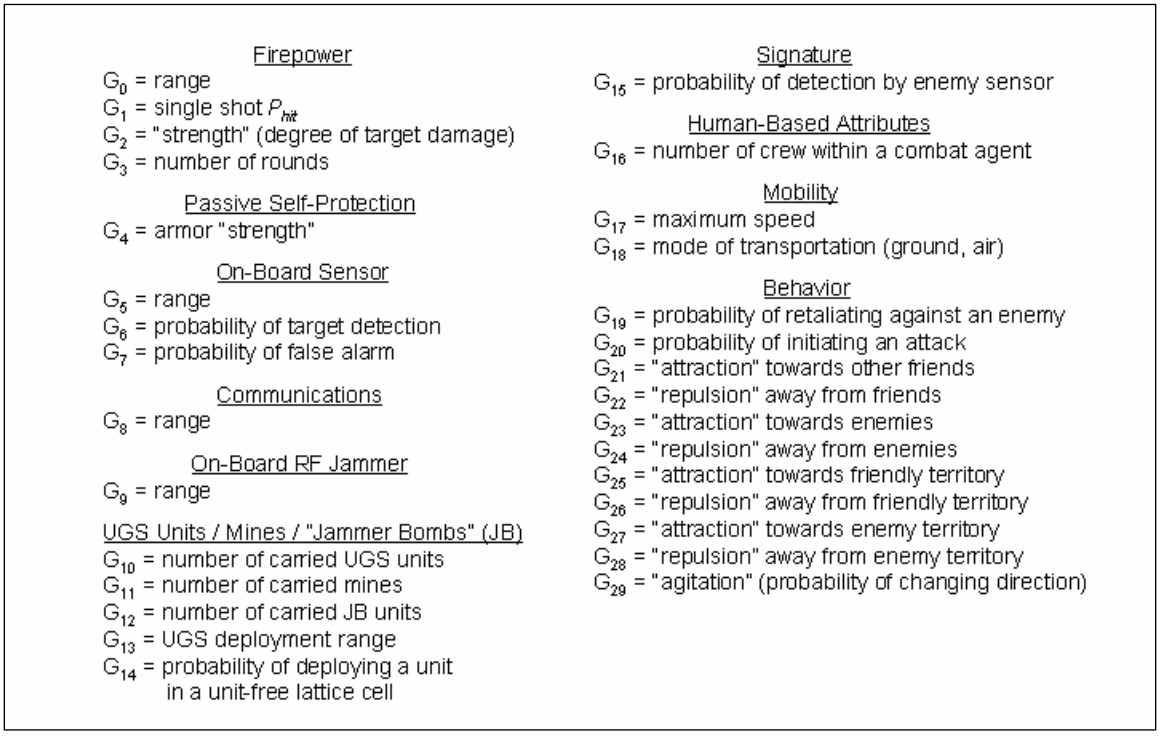


Figure 8. The genetic schema representing hardware and behavioral attributes of a generic combat agent.

2.3 Combat Agent Dynamics

Once the genes within a generic combat agent schema have been specified, the rules governing how these genes are utilized by combat agents within a dynamic CA simulation can be delineated. In the following subsections, these CA transition rules (involving chromosome genes within the capability, signature, and behavioral classes) are discussed in detail.

2.3.1 Firepower

The firepower weapon maintained by a combat agent is defined by maximum weapon range (gene G_0), single-shot probability of hitting an engaged target (gene G_1), fired projectile "strength" (gene G_2), and the number of rounds remaining of the initial number of rounds (gene G_3). The weapon dynamics are described by the following pseudocode:

IF target range $\leq G_0$ AND *random*%1.00 < G_1 AND remaining rounds > 0 THEN

remaining rounds = remaining rounds – 1

IF target armor strength < G_2 THEN

target is killed

ELSE IF target armor strength $\geq G_2$ THEN

target armor strength = target armor strength – G_2

target survives encounter

END

ELSE

weapon is not fired

END.

Here, *random* is a random whole number, so that *random*%1.00 represents a random fraction sampled from the range [0, 1.0]. In other words, the firing combat agent will kill an engaged target based on a probability of hit $P_{hit} = G_1$ given that (a) the target is within firing range, (b) the engaging combat agent has at least one remaining round left, and (c) the target's armor strength is less than the engaging weapon strength G_2. In all other cases, the target will survive the encounter.

2.3.2 Passive Self-Protection

Combat agents are self-protected by conventional passive (as opposed to active) armor characterized by the initial armor strength G_4. Then, upon engagement, the combat agent's armor strength is reduced by the engaging agent's firepower strength until the residual armor strength < 0 (at which point, an engaged combat agent "dies").

2.3.3 On-Board Sensor

The single on-board sensor (representing some type of notional multimodal sensor suite) maintained by a combat agent is defined by maximum sensor range (gene G_5), probability of target detection (i.e., the likelihood of sensing a neighboring agent when it is present [gene G_6]), and probability of false alarm (i.e., the likelihood of reporting an agent's presence when it is really *not* present [gene G_7]). In addition, if the neighboring combat agent is an enemy of the sensing agent, then the acquisition signature of the target agent (gene G_{15} [see section 2.3.7]) is also utilized. During a simulation time cycle, each combat agent that remains "alive" or functional sends out a "health status" message reporting its CA lattice location and identity (i.e., Blue or Red). This status message is then "sensed" by a sensor-equipped combat agent as described by the following pseudocode:

IF distance to sensed entity $\leq G_5$ AND *random*%1.00 < G_6 AND ((sensed entity is an enemy of sensing agent AND *random*%1.00 < G_{15}) OR sensed entity is an ally of sensing agent) THEN

acquire "health status" message from sensed entity

END.

Again, *random*%1.00 represents a random fraction sampled from the range [0, 1.0] so that an entity is sensed with probability of detection $P_{detect} = G_6$ given that it is within the sensing agent's sensor range.

At the same time, the sensing combat agent's sensor may generate a false positive as described by the following:

IF $G_5 > 0$ AND *random*%1.00 < G_7 THEN

acquire self-generated false "health status" message with randomly selected CA lattice location and sensed entity identity

END.

Here, the sensor will generate a false "health status" message (that the sensing agent believes to be correct data) with probability of false alarm $P_{false\ alarm} = G_7$ given that the sensor range > 0.

2.3.4 Communications

Combat agent intercommunication is defined by the maximum communication range (gene G_8). This range reflects the transmitter power limitations of a message-transmitting combat agent so that a receiving combat agent will be able to accept a transmitted message given that (a) the receiver lies within the extended Moore CA neighborhood of size $(2*G_8 + 1) \times (2*G_8 + 1)$ cells relative to the transmitter agent and (b) the receiving agent is not currently being jammed. The communication capability is employed only between allied combat agents in order to share SA data messages (which report the identities and positions of both friendly and enemy agents) acquired by sensor-equipped agents.

2.3.5 On-Board RF Jammer

Combat agent on-board communications jamming capability is defined by the maximum jamming range (gene G_9). As with communication, this range reflects the RF power limitations of a jamming combat agent, so that a receiving combat agent will *not* be able to accept messages from another transmitting agent given that the receiver lies within the extended Moore CA neighborhood of size $(2*G_9 + 1)$ by $(2*G_9 + 1)$ cells relative to the jamming agent. This notional on-board jammer is assumed to be able to disrupt transmissions equally across all RF communication bands.

2.3.6 Supplies of UGS Units/Mines/Jammer Bombs

The supply and deployment of UGS units, land mines, and RF jammer bombs carried by a combat agent are defined by the numbers of carried UGS units, land mines, and jammer bombs (the initial numbers of which are encoded in genes G_{10}, G_{11}, and G_{12}, respectively), maximum UGS deployment range (gene G_{13}), and the probability of deploying a UGS/mine/jammer bomb unit in a currently unit-free lattice cell (gene G_{14}). For land mines and jammer bombs, the maximum deployment range is assumed to be equal to zero (i.e., a combat agent can only emplace a mine or jammer bomb into the lattice cell the agent currently occupies, where the mine or bomb will remain after the agent chooses to move on at some future time). The overall deployment process is described by the following pseudocode:

IF number of remaining UGS units > 0 AND *random*%1.00 < G_{14} THEN

number of remaining UGS units = number of remaining UGS units – 1

place a UGS unit into a cell if the distance to the target cell $\leq G_{13}$

END.

IF number of remaining land mines > 0 AND *random*%1.00 < G_{14} THEN

number of remaining land mines = number of remaining land mines – 1

place a land mine into the current cell

END.

IF number of remaining jammer bombs > 0 AND *random*%1.00 < G_{14} THEN

number of remaining jammer bombs = number of remaining jammer bombs – 1

place a jammer bomb into the current cell

END.

It should be noted that although the probability of deploying a unit into a unit-free cell is the same for UGS, land mine, and jammer bomb units, the decisions to deploy each type of unit are executed independently of each other. Also, it is assumed that UGS units, land mines (which impact combat agents only), and jammer bombs (which do not affect a UGS unit's sensing capability) will not functionally interfere with each other if placed within the same cell.

In the following subsections, the specific functional dynamics of UGS units, land mines, and RF jammer bombs are described in detail.

2.3.6.1 UGS Units

Within the context of the current CA model, a UGS unit is a set of fields that is attached to a cell, representing the placement of a notional autonomous sensor onto a ground location within the battlespace (Followill et al. 1997; Haider 1998). A combat agent transmits a UGS unit to a target cell by sending a "UGS missile" message to the cell. This message represents a notional missile

packed with one or more individual UGS units on board, which are dispersed to fill a neighborhood surrounding the target cell once the "UGS missile" message reaches its target. Then, upon reaching a cell (or neighborhood of cells), the "UGS missile" message maps the UGS fields *UGS range*, *UGS owner* (i.e., Blue or Red), and *UGS transmit/receive functionality* (for internetted data sharing) into that cell. As discussed in section 2.3.6, the size of the extended Moore CA neighborhood that a combat agent may fire "UGS missile" messages to is defined by chromosome gene G_{13}. The size of the neighborhood surrounding a UGS target cell, however, is preset within the CA code (see section 3.2.3 for an example of UGS deployment via "UGS missile" messages).

Once a UGS unit has been positioned in a CA lattice cell (where it will remain for the duration of a simulation run), the unit begins to function as a remote sensor. Figure 9 illustrates the operational dynamics of a UGS unit. In Figure 9a, the UGS unit senses its local environment, which consists solely of the lattice cell it occupies. Then, in Figure 9b, as a Red combat agent moves into the cell and is sensed by the UGS unit, an SA data message that reports the x/y lattice position and identity (i.e., Blue or Red) of the sensed agent is transmitted to the adjacent north/south/east/west cells making up a von Neumann CA neighborhood. Finally, if these adjacent cells are occupied by allied internetted UGS units that are currently unjammed, the messages are received and retransmitted across new von Neumann neighborhoods surrounding the relaying UGS units. In this manner, SA reports can propagate across a network or "web" of UGS units, thus allowing sensor-equipped combat agents with limited sensor range to access SA information from local nodes within the UGS web.

2.3.6.2 Land Mines

Unlike UGS units, land mines can only be deployed into the lattice cell currently occupied by a dispensing combat agent. This is done by directly attaching the characteristic land mine parameter field *mine owner alliance* (i.e., Blue or Red) to the cell occupied by the mine. This parameter serves a twofold purpose—it identifies the battlefield alliance of the dispensing agent (1) for combat agent damage purposes (see the following) and (2) to prevent other agents from the same alliance from dropping additional mines into the cell. Once deployed, the land mine dynamics are described by the following pseudocode:

> IF a combat agent enters the land mine cell AND combat agent alliance ≠ mine owner alliance AND *random*%1.00 < 0.5 THEN
>
> Reset agent chromosome gene G_{17} (maximum speed) = 0
>
> END.

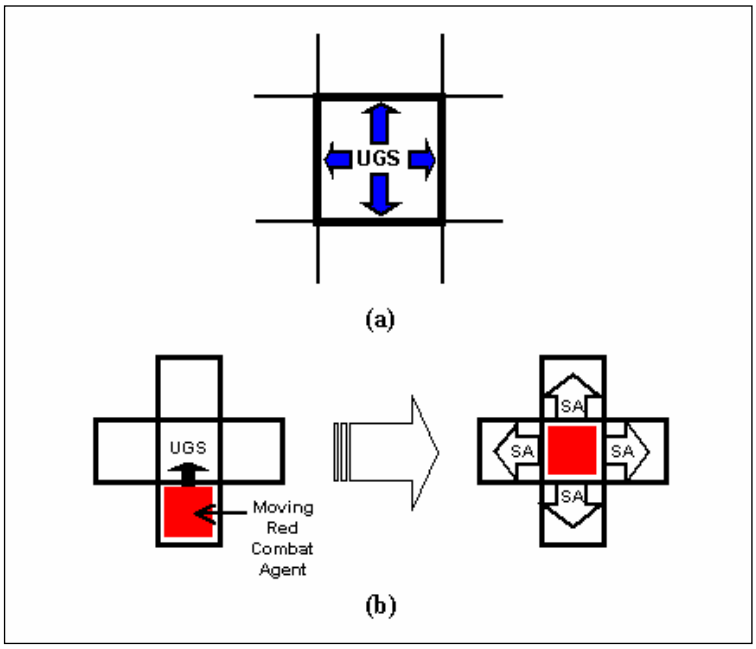


Figure 9. Operational dynamics of a UGS unit: (a) UGS unit senses its local environment and (b) UGS unit transmits an SA report on the sensed agent across a von Neumann CA neighborhood.

Thus, if a combat agent enters a lattice cell also occupied by a land mine previously planted there by another agent belonging to the enemy alliance relative to the first agent, there is a 50% probability that the entering combat agent will lose its mobility capability (by setting $G_{17} = 0$ within the agent's chromosome). Here, the assumption is made that combat agents represent mobile hardware platforms as opposed to infantry so that land mines result in degraded agent capability as opposed to death. It is also assumed that a land mine will not affect the mobility of agents belonging to the same battlefield alliance responsible for first planting the mine. Finally, it is assumed that each land mine is actually a cluster of bomblets spread across a lattice cell so that a land mine continues to function (essentially perpetually) after multiple detonations.

2.3.6.3 RF Jammer Bombs

As with land mines, RF jammer bombs can only be deployed into the lattice cell currently occupied by a dispensing combat agent. Again, this is done by directly attaching the characteristic jammer bomb parameter fields *bomb owner alliance* and *total jammer bomb lifetime* (in simulation time cycles) to the cell occupied by the bomb. The first bomb parameter is used to prevent other members of the same alliance as the bomb-dispensing agent from dropping additional jammer bombs into the cell, while the second parameter is used to set the lifetime of the battery powering the jammer bomb (which is decremented by 1 once every simulation time cycle). Once deployed, a jammer bomb acts to jam the communication reception

capability of *all* combat agents (irregardless of battlefield alliance) and internetted UGS units within a predefined extended Moore CA neighborhood given that the residual bomb's lifetime > 0. Figure 10 depicts the jamming neighborhood of a jammer bomb whose jamming range = 2 (this parameter can be adjusted within the CA code).

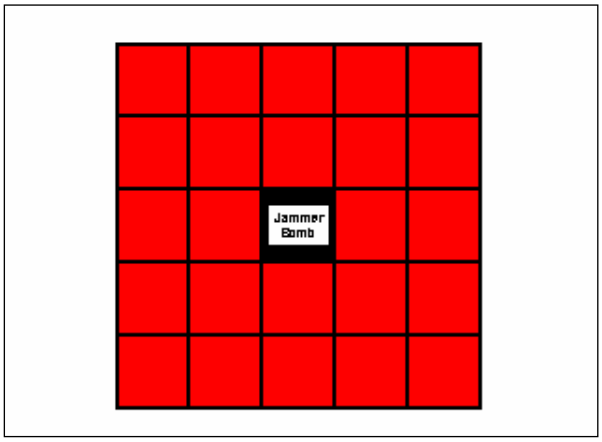


Figure 10. The extended Moore CA neighborhood surrounding a notional RF jammer bomb.

2.3.7 Signature

The acquisition signature of a combat agent is represented as the probability that the agent will be detected by an enemy combat agent equipped with an on-board sensor (gene G_{15}). As described in section 2.3.3, this gene works in conjunction with a sensor-equipped enemy agent to determine whether the identity and position of the sensed combat agent is indeed acquired. It should be noted that, since UGS units are constrained to sense only a very limited local environment (i.e., a single lattice cell), these types of sensors are assumed to acquire targets irregardless of the target agent's signature gene allele value.

2.3.8 Human-Based Attributes

Within the context of combat agents, the functional role of human soldiers (which can be thought to occupy a combat agent platform) is simply to constrain the engagement opportunities of "unmanned" combat agents. This type of constraint is implemented via gene G_{16}, which encodes the number of human crews within a combat agent. Basically, the constraint allows an unmanned agent (where $G_{16} = 0$) to fire upon an acquired target *only* when the target SA data has been supplied to the unmanned agent by another allied agent with one or more human crews (where, for the latter agent, $G_{16} > 0$). Although meant to suggest human-based control of all target engagements, the cognitive capabilities of combat agents in the current model are purely *reactive* in nature. Thus, the only impact of gene G_{16} on combat unit effectiveness is either to (a) force the unit to rely heavily on its allied communication network (when most agents in the unit are unmanned) or (b) force the unit configuration to evolve to a point where most agents are manned (and thus operate autonomously with minimal interagent cooperation).

2.3.9 Mobility

Combat agent mobility is characterized by a maximum allowable speed (gene G_{17}) and mode of transportation (gene G_{18}). Since combat agents are allowed to move (either north, south, east, or west) one cell length per simulation time cycle at most, relative agent speed reflects the probability that an agent will move one cell length during that time cycle. Thus, the probability of agent movement during the n^{th} simulation time cycle is

$$P_{move}(t_n) = \frac{|v(t_n)|}{G_{17}}, \tag{1}$$

where $|v(t_n)|$ is the agent "speed" (the magnitude of the velocity v) during the n^{th} time cycle. Possible modes of transportation include (a) ground, wherein agents must move around "terrain impediment" cells added to the CA lattice (representing impassible mountains, rivers, and buildings), and (b) air, wherein agent movement is unrestricted by terrain. Combat agent movement is then described by the following pseudocode:

IF $G_{17} > 0$ AND *random*%1.00 < current agent speed/G_{17} THEN

    IF G_{18} = ground THEN

        IF terrain in direction of intended movement is passable THEN

            move one cell in direction of intended movement

        ELSE IF terrain is impassible THEN

            rotate 180° and then move in that new direction

        END

    ELSE IF G_{18} = air THEN

        move one cell in direction of intended movement

    END

ELSE

    remain in the current cell

END.

Calculation of the direction of intended combat agent movement is described in detail in the next section.

2.3.10 Behavior

Combat agent behavior is encoded into a vector within the agent chromosome representing the "personality" of that agent. In the following subsections, modes of combat agent behavior that define combat engagement actions, goal-driven mobility, and agitation-driven mobility are discussed.

2.3.10.1 Combat Engagement Actions

Engagement actions demonstrated by a combat agent are characterized by (a) the probability that the combat agent will initiate a firepower attack against an enemy agent P_{attack} (gene G_{20}) and (b) the probability that the agent will retaliate against an enemy once attacked $P_{retaliate}$ (gene G_{19}). The application of P_{attack} and $P_{retaliate}$ to instantiating a combat action is described by the following pseudocode:

> IF G_0 (firepower range) > 0 AND number of remaining firepower rounds > 0 AND a combat agent has acquired and selected a target to engage THEN
>
> > IF distance between engaging combat agent and target $\leq G_0$ AND *random*%1.00 < P_{attack} THEN
> >
> > > fire one round at target
> >
> > END
>
> IF agent has just received fire AND distance between engaged agent and attacking agent $\leq G_0$ AND *random*%1.00 < $P_{retaliate}$ THEN
>
> > fire one round at attacking agent
>
> END

END.

Here, $G_0 = 0$ indicates a combat agent without firepower capability, and the number of remaining firepower rounds $\leq G_3$. Also, a combat agent will select to engage the first target agent it acquires (i.e., the agent reported in the first queued SA message within the agent's internal message database) provided that (a) the target is within engagement range and (b) the engaging agent *perceives* the target agent as an enemy.

2.3.10.2 Goal-Driven Mobility

Combat agent mobility is motivated by spatially-oriented goal points towards which agents are coded to either approach or avoid. These goal points are of two basic types: (1) the center of mass (CM) position of a cluster of combat agents and (2) a mission-related territorial objective

location within the CA lattice. Here, the first goal point type is dynamic (i.e., a function of collective combat agent speed and direction), whereas the second type is preassigned to a lattice cell (or a local cluster of contiguous cells) and thus remains static throughout a simulation.

In movement motivated by the first type of mobility goal point, a combat agent is either attracted, repulsed, or both attracted *and* repulsed by the CM of the unit to which the agent belongs (i.e., the friendly combat unit) as well as the CM of the enemy force (i.e., the foe combat unit). An agent's attraction to and repulsion from the friendly unit CM is based on weighting factors (within the range [0, 100]) that are stored in chromosome genes G_{21} and G_{22}, respectively. Likewise, the agent's attraction to and repulsion from the foe unit CM is based on similar weighting factors that are stored in chromosome genes G_{23} and G_{24}, respectively. Employing these attraction/repulsion weights as encoded in genes G_{21} through G_{24} of the i^{th} combat agent's chromosome, the x and y lattice coordinates of the superimposed friendly and foe CMs relative to that agent are as follows:

$$CM_x^{agent_i}(t_n) = \left(G_{21}^{agent_i} - G_{22}^{agent_i}\right) * \sum_{friends} x_{friend}(t_n) - x_{agent_i}(t_n) + \left(G_{23}^{agent_i} - G_{24}^{agent_i}\right) * \sum_{foes} x_{foe}(t_n) - x_{agent_i}(t_n), \quad (2)$$

and

$$CM_y^{agent_i}(t_n) = \left(G_{21}^{agent_i} - G_{22}^{agent_i}\right) * \sum_{friends} y_{friend}(t_n) - y_{agent_i}(t_n) + \left(G_{23}^{agent_i} - G_{24}^{agent_i}\right) * \sum_{foes} y_{foe}(t_n) - y_{agent_i}(t_n), \quad (3)$$

respectively, where $x_{friend}(t_n)$ and $y_{friend}(t_n)$ are the x and y lattice coordinates at the n^{th} time cycle, respectively, of a neighboring agent friendly to the i^{th} agent; $x_{foe}(t_n)$ and $y_{foe}(t_n)$ are the x and y lattice coordinates at the n^{th} time cycle, respectively, of a neighboring agent hostile towards the i^{th} agent; and $x_{agent_i}(t_n)$ and $y_{agent_i}(t_n)$ are the x and y lattice coordinates at the n^{th} time cycle, respectively, of the i^{th} agent itself. Thus, equations 2 and 3 calculate the net combined friendly/foe combat unit CM location relative to the i^{th} agent by summing over the difference in position of all other friendly and enemy agents from the position of the i^{th} agent, multiplying these sums by the *net* attraction (which can be less than 0, indicating a repulsion) of the i^{th} agent towards friendly and foe unit CMs, and then adding the results.

In movement motivated by territorial objective goal points, a combat agent is either attracted, repulsed, or both attracted and repulsed by a mission-related territorial importance gradient that defines the relative mission importance of a lattice cell (or local cluster of contiguous cells) relative to a predefined objective point within the lattice. In the current model implementation, an importance gradient is assigned to territory "owned" or controlled by either a Blue or Red

combat unit (i.e., Blue or Red territory, respectively), where each gradient has a maximally-important lattice location representing combat unit "headquarters" towards which an enemy agent might want to approach (or avoid) and a set of contiguous minimally-important locations representing unclaimed neutral territory. As with movement driven by combat unit CMs, a combat agent's attraction to and repulsion from both the friendly and foe headquarter locations (i.e., the mission objective points) is based on weighting factors that are stored in chromosome genes G_{25} through G_{28}, respectively. Employing these attraction/repulsion weights, the *x* and *y* lattice coordinates of a new type of CM based on the i^{th} agent's reaction to local territorial subgradients within both friendly and foe importance gradients are as follows:

$$CM_x^{territory_i}(t_n) = \sum_{neighborhood(t_n)} \begin{cases} \left(G_{25}^{agent_i} - G_{26}^{agent_i}\right) * Importance_{cell} * \left(x_{cell} - x_{agent_i}(t_n)\right) \\ \quad when\ cell.owner\ =\ agent.alliance \\ \left(G_{27}^{agent_i} - G_{28}^{agent_i}\right) * Importance_{cell} * \left(x_{cell} - x_{agent_i}(t_n)\right) \\ \quad when\ cell.owner\ =\ !agent.alliance \end{cases}, \quad (4)$$

and

$$CM_y^{territory_i}(t_n) = \sum_{neighborhood(t_n)} \begin{cases} \left(G_{25}^{agent_i} - G_{26}^{agent_i}\right) * Importance_{cell} * \left(y_{cell} - y_{agent_i}(t_n)\right) \\ \quad \text{when cell.owner = agent.alliance} \\ \left(G_{27}^{agent_i} - G_{28}^{agent_i}\right) * Importance_{cell} * \left(y_{cell} - y_{agent_i}(t_n)\right) \\ \quad \text{when cell.owner = !agent.alliance} \end{cases}, \quad (5)$$

respectively, where $Importance_{cell}$ is the local value of a territorial importance gradient within a lattice cell (i.e., an integer ranging from 0 up to 128), x_{cell} and y_{cell} are the *x* and *y* lattice coordinates, respectively, of the lattice cell under examination, and $xagent_i(t_n)$ and $y_{agenti}(t_n)$ again are the *x* and *y* lattice coordinates at the n^{th} time cycle, respectively, of the i^{th} agent. Here, equations 3a and 3b calculate the superposition of the friendly and foe local importance subgradient CM locations relative to the i^{th} agent by multiplying the difference in position of neighboring subgradient cells from the position of the i_{th} agent by the importance of each subgradient cell and the net attraction of the i^{th} agent towards friendly and foe subgradient CMs and then summing these results over all cells within a preset subgradient neighborhood. An important distinction is made between subgradient cells owned by the i_{th} agent's friends (in which case, the logical relation cell.owner = agent.alliance evaluates to TRUE) vs. cells owned by foes (in which case, the logical relation cell.owner ! = agent.alliance evaluates to TRUE).

Once the agent cluster and importance subgradient CM locations have been calculated for the ith agent, they are combined into a total net CM location according to the following:

$$CM_x^{total_i} = \alpha * CM_x^{agent_i} + \beta * CM_x^{territory_i}, \quad (6)$$

and

$$CM_{y}^{total_i} = \alpha * CM_{y}^{agent_i} + \beta * CM_{y}^{territory_i}, \quad (7)$$

where α and β are weighting constants representing the relative degree to which the i^{th} agent desires to move toward (or avoid) friendly/enemy agent clusters vs. friendly/enemy territorial objective points.* Thus, $CM_{x}^{total_i}$ and $CM_{y}^{total_i}$ provide the x and y coordinates, respectively, towards which the i^{th} agent intends to move during the next simulation time cycle.

In addition to deciding upon a next direction of travel, a combat agent will also adjust its speed of movement relative to the total CM coordinates described in equations 6 and 7 (where this net CM is a function of both dynamic agents and static territory). This speed adjustment is set as a function of the "acceleration" of the i^{th} agent at the n^{th} time cycle

$$\Delta speed_{agent_i}(t_n) = \frac{CM_{x}^{total_i} * CM_{y}^{total_i}}{t_n - t_{n-1}} - \left| velocity_{agent_i}(t_n) \right|^2, \quad (8)$$

where $t_n - t_{n\text{-}1}$ is the duration of one simulation time cycle (i.e., one unit) and $|velocity_{agent_i}(t_n)|$ is the magnitude of the agent velocity (i.e., the speed) at the n^{th} time cycle. Since agent speed is more akin to the probability of moving a fixed distance per time step rather than a true physical speed, the "acceleration" proposed in equation 8 is actually an artificial construct that can be used to maintain the position of the i^{th} agent as close as possible to the total CM location throughout a simulation (i.e., by adjusting agent speed so that $\left(CM_{x}^{total_i} * CM_{y}^{total_i}\right) / \left(t_n - t_{n-1}\right)$ $= |velocity_{agent_i}(t_n)|^2$ or $\Delta speed_{agent_i}(t_n) = 0$). Thus, the new speed of the i^{th} agent at the $(n+1)^{th}$ time cycle is given by the following:

$$speed_{agent_i}(t_{n+1}) = \begin{cases} speed_{agent_i}(t_n) + 1 \text{ when } \Delta speed_{agent_i}(t_n) \geq \Delta speed_{min} \\ \quad \text{and } speed_{agent_i}(t_n) < speed_{max} \\ speed_{agent_i}(t_n) - 1 \text{ when } \Delta speed_{agent_i}(t_n) \leq -\Delta speed_{min} \\ \quad \text{and } speed_{agent_i}(t_n) > 0 \end{cases}, \quad (9)$$

where $\Delta speed_{\min}$ defines a minimal threshold value of $\Delta speed_{agent_i}(t_n)$ that triggers a speed adjustment when achieved or surpassed and $speed_{\max}$ is the maximum allowable agent speed. Both $\Delta speed_{\min}$ and $speed_{\max}$ are adjustable but must remain fixed upon commencement of a simulation run. Interestingly, the algorithms described in equations 2–9 model a variety of agent "flocking" behavior that was first demonstrated by Reynolds in his Boids model (Reynolds 1987).

* In the current model implementation, these weights are fixed at $\alpha = \beta = 0.5$, representing equal prioritization between agent- and territorial-oriented mobility goals.

2.3.10.3 Agitation-Driven Mobility

In addition to the goal-motivated movement discussed in section 2.3.10.2, combat agents can also be driven to move by "agitation." An agitated agent is thus characterized by $P_{agitation}$, the probability that the agent will randomly select and pursue a new direction of travel (while maintaining the same speed as calculated using equations 8 and 9) per simulation time cycle (gene G_{29}). The application of $P_{agitation}$ to instantiate random movement within a combat agent is described by the following pseudocode:

IF *random*%1.00 < $P_{agitation}$ THEN

    new direction = *random*%4

IF new direction = 0 THEN

    move north

ELSE IF new direction = 1 THEN

    move east

ELSE IF new direction = 2 THEN

    move south

ELSE IF new direction = 3 THEN

    move west

END

END.

It should be noted that an agitated agent will revert back to the nonagitated state at the start of each new time cycle (and will again move randomly based on $P_{agitation}$), and thus the behavioral dynamics of agitated movement is modeled as a random process with no memory.

---

3. Combat Agent Simulations

Now that the elements of the CA simulation engine have been described, the development and demonstration of a software framework that utilizes the CA engine for IO-stressed Blue/Red combat simulations can be discussed. Due to the prolonged runtimes necessary to simulate densely-packed UGS networks and multihop message relaying within the CA implementation of combat agent communication nets, the decision was made to codevelop two different versions of the IO simulation software framework. In the first framework (section 3.1), combat-motivated coevolution of both Blue and Red unit configuration genomes is simulated within an IO-stressed

virtual environment. In this framework, the duration of an update time cycle within the CA simulation engine is compressed to facilitate the thousands of Blue/Red combat instances necessary to execute the genomic coevolution. In the second framework (section 3.2), the CA engine time cycle duration is expanded to allow for the simulation of UGS networks and multihop message relaying previously mentioned.

3.1 Blue/Red Unit Combat-Driven Coevolution

3.1.1 Genomic Coevolution Framework

The first software framework was developed in order to explore the dynamics of two opposing but coevolving units of networked autonomous combat agents (Figure 11). Within this framework, "friendly" (Blue) and "foe" (Red) agents are encoded within the combat unit configuration genome introduced in section 2.2.2, where the metrics within a genome represent notional agent hardware-based capabilities and combat-oriented behaviors. In this context, the genome represents an instantiation of a possible configuration of networked mobile platforms making up a network-centric combat unit. Initial populations of Blue/Red candidate genomes (or candidate combat unit configurations) are randomly generated by the random_genome routine working in combination with a cost filter (see section 3.1.5). These candidate genomes are then input (via the place routine) into the `combat_ga` CA simulation engine, along with a virtual landscape (generated by the terrain routine) and a territorial importance gradient (generated by the territory routine), and a combat simulation is executed. Such combat simulations are subsequently run for each possible Blue/Red genome pairing (or a subset thereof).

During a Blue/Red combat simulation, a data filter dynamically calculates the dynamic operational "fitness" (analogous to one or more combined combat unit measures of performance averaged over a combat mission time interval) of each opposing combat agent unit within the simulation (the functions used to calculate dynamic fitness are described in detail in section 3.1.4). These combat simulations can include hostile IO actions (which, in this case, are limited to RF jamming by jammer-equipped combat agents). The output of the data filter is then either directed into a GA (i.e., the `reproduce` routine) or can be directly analyzed in time series format. In the former case, the top five "fittest" genomes for each side are determined (based on the collective results from all instantiated Blue/Red combat simulations) and then used by the GA to create new "offspring" genome populations (through the use of algorithmic "genetic operations" performed on genome pairs [section 3.1.4] working in combination with the cost filter previously mentioned). These replenished populations are used to continue the coevolutionary process across successive genome "generations;" this process is iteratively repeated until both Blue and Red genome fitness values converge at or near a maximal value. The genome populations can also be dynamically classified (as a function of hardware-based capabilities and behaviors encoded into a genome) throughout a coevolutionary simulation run by the classify routine (for a demonstration of this routine, see section 3.1.6.2).

The Blue/Red coevolutionary process described in the previous paragraphs is executed by running the Evolve Perl script. There are two other available routines for running coevolution simulations that are functionally redundant with Evolve (indicated by the "||" symbol at the top of Figure 11) but can also take advantage of parallel processing computer hardware capability when available. The first of the redundant routines is parAvgEvolve, a Perl script identical to Evolve except that each simulation instance of a Blue/Red genome pairing is partitioned and run on different parallel processor threads. Then, results from multiple instances of the genome pairing are averaged in parallel. The second of the redundant routines is parGenEvolve, another Perl script identical to Evolve except that now a sequence of nonpartitioned simulation instances of a specific Blue/Red genome pairing is assigned to a processor, and then instance sequences covering different genome pairings are run in parallel. After that, the results from each sequence run are averaged on the respective processor, and averaged results from all genome pairings are centrally collected.

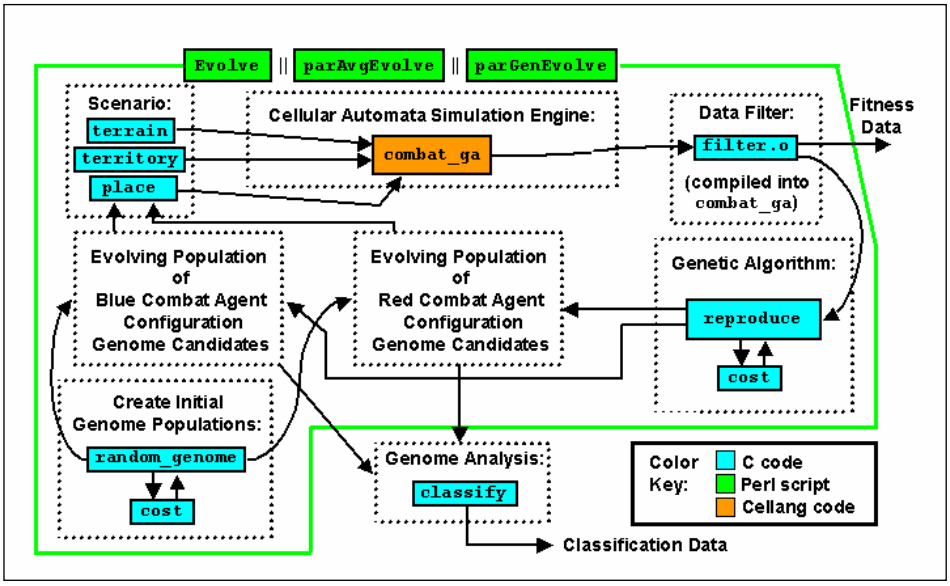


Figure 11. Software framework for simulating Blue/Red combat unit genomic coevolution.

3.1.2 The `combat_ga` CA Simulation Engine

Written in Cellang (see section 2.1), the `combat_ga` CA simulation engine (so named to indicate its specific design for implementation within the GA-based coevolutionary simulation framework) models the interaction between two opposing units of combat agents. The operational dynamics of this simulation engine are cyclic in design, where a repetitive sequence of actions is executed in parallel for each cell within the CA universe during each successive simulation time cycle. This sequence is depicted in Figure 12, which illustrates the following cycle of processes (where a process can unfold over one or more sequential simulation time steps):

- Combat agents move into an adjacent cell within their local von Neumann neighborhood based on their goal-driven movement decisions (see section 2.3.10.2). This process is executed over one time step.
- Combat agent communication reception is jammed by neighboring enemy agents equipped with on-board RF jamming capability (i.e., gene G_9 within the enemy agent chromosome > 0 [see section 2.3.5]). This process is executed over one time step, which runs concurrently with the first time step of the subsequent process discussed next.
- Messages transmitted from combat agents to neighboring agents are propagated across the CA universe. These messages can include (a) acquired data traveling from sensed entities to sensor-equipped agents, (b) sensor data shared amongst allied agents over their communication network, and (c) firepower payloads moving from an engaging agent towards an acquired target. This process is executed over nine sequential time steps.
- Engaged targets react to received fire (i.e., either die or survive engagement with reduced armor capability). This last process is executed over one time step.

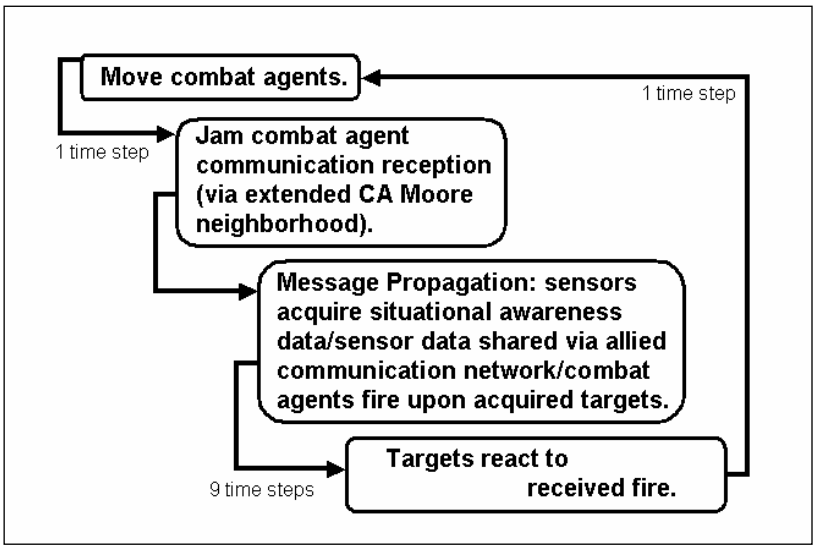


Figure 12. Cyclic dynamics of the `combat_ga` CA simulation engine.

Thus, one entire cycle of simulation processes unfolds over 11 sequential time steps, which are iteratively repeated throughout a combat simulation instance.

The third process or phase within the `combat_ga` simulation time cycle depicted in Figure 12 involves the propagation of messages between stationary combat agents positioned in cells located anywhere across the CA lattice. In order to minimize the duration of the message propagation phase, an intelligent relay network (IRN) algorithm was developed to handle message delivery to receiving agents. This algorithm essentially builds a series of relay stations into the underlying fabric of the CA universe, where the relays are assigned to cells and are distributed in a regular and uniform manner on a rectangular grid throughout the CA lattice.

The collective relay network is "intelligent" in that it guarantees that each cell within the CA lattice only receives one copy of any particular message by the end of the message propagation phase.

Figure 13 provides a demonstration of an IRN implemented within a limited 12 × 12 cell CA lattice. In this implementation, the lattice is partitioned into 16 nonoverlapping Moore neighborhoods, where the center cell within each neighborhood functions as a local relay station.

In the northwest quadrant of the CA lattice is a transmitter combat agent (represented by a "T") that launches a message intended for all neighboring receiver agents (represented by an "R"). This message is disseminated to all receiving agents in five successive phases. First, from time steps t_n to t_{n+1}, a copy of the message is transmitted *horizontally* from the T cell to the center cell of the right-hand adjacent Moore neighborhood. Next, from time steps t_{n+1} to t_{n+2}, copies of the message are again transmitted horizontally from the T cell and the previous message destination site to the center cells of the left- and right-hand adjacent Moore neighborhoods, respectively. In the third phase, from time steps t_{n+2} to t_{n+3}, copies of the message are transmitted *vertically* downward from the center cells within the four Moore neighborhoods in the second row (each of which contains either the original message or a copy thereof) to the center cells of adjacent lower Moore neighborhoods. Following this, from time steps t_{n+3} to t_{n+4}, copies of the message are transmitted both vertically upward and downward to the center cells of adjacent Moore neighborhoods. Finally, from time steps t_{n+4} to t_{n+5}, copies of the message are transmitted from the center cells within all 16 Moore neighborhoods (each of which again contains either the original message or a copy thereof) to those intraneighborhood cells that contain receiver agents. In general, given an IRN consisting of M symmetric neighborhoods (where M must be an *even* number) with N cells per neighborhood (and thus a symmetric CA lattice with $M*N$ total cells)* the following occurs:

- the first "horizontal transmission" and second "vertical transmission" phases unfold over $M^{1/2}/2$ time steps, and
- the third and final "intraneighborhood transmission" phase unfolds over one time step.

Given that the IRN within the `combat_ga` simulation engine was designed with $M = 64$ (8 horizontal × 8 vertical neighborhoods) and $N = 289$ (a symmetric neighborhood of 17 horizontal × 17 vertical cells), the entire message propagation process unfolds over $M^{1/2} + 1 = 9$ time steps.

* The symmetric neighborhoods depicted in Figure 13 happen to be of the nine-cell Moore type but can be made larger as the size of the CA lattice increases. The downside of this is that the degree of maximally allowed CA lattice parallelization scales inversely with neighborhood size N so that $P \propto N^{-1}$, where P = maximum number of parallel computational threads the CA lattice may be assigned to.

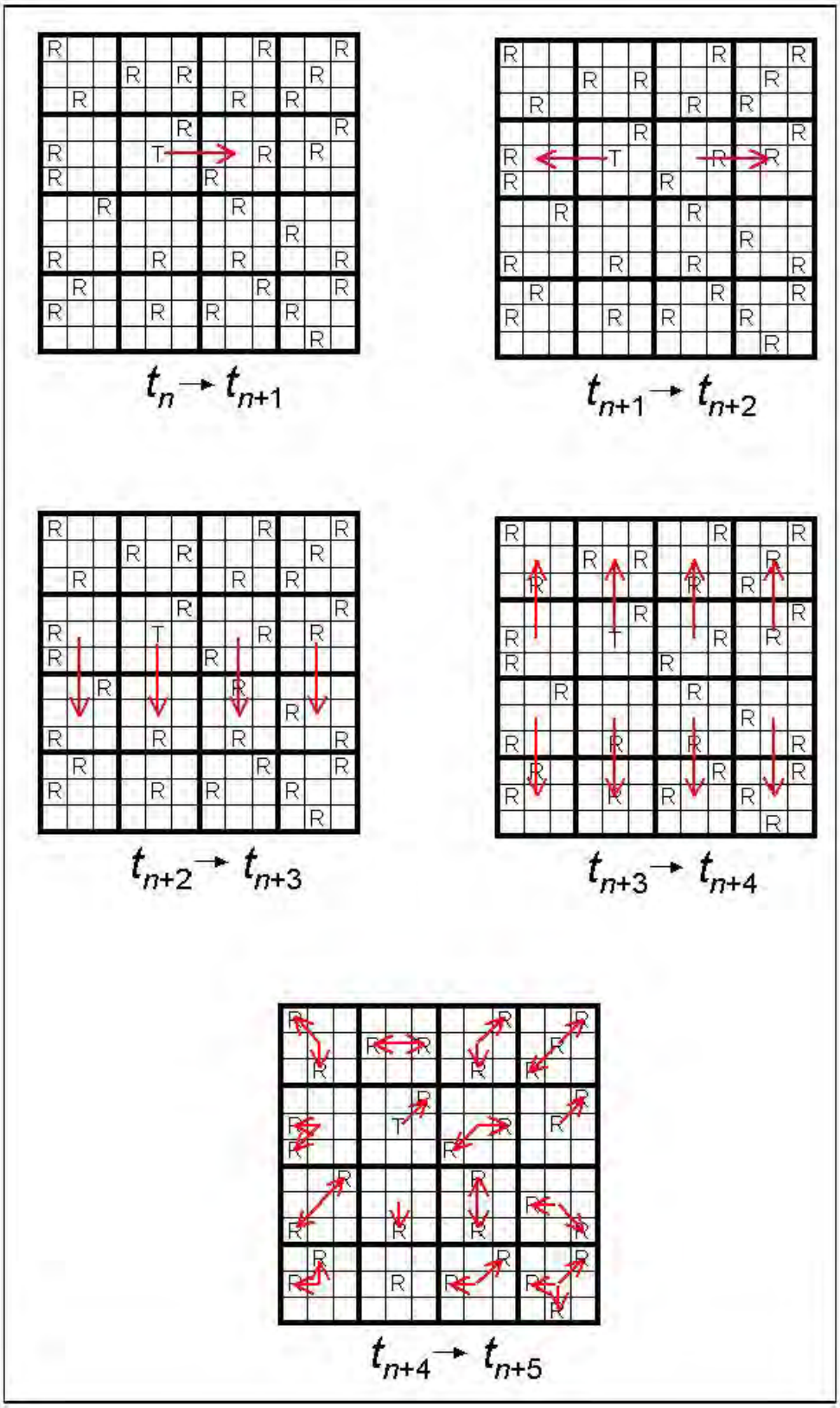


Figure 13. Demonstration of the intelligent relay network as implemented within `combat_ga`.

A subtle point that needs to be further discussed is the superposition of different processes that occur during the message propagation phase depicted in Figure 12. There is an inherent sequential causality in the three message propagation processes, where (1) propagation of signals from sensed entities to sensor-equipped combat agents is followed by (2) sensor data sharing over a communication net, and finally (3) the exchange of fire between engagers and targets. Since the `combat_ga` CA engine was purposefully designed to minimize simulation run times, a causal sequence of events involving processes 1–3 can actually unfold over three successive time cycle updates. Since combat agents are allowed to move one cell at the start of every time cycle, an engaged target could be two cells removed from the location fired at by an agent acting on data received during a previous time cycle. To work around this problem, tags are used to link fired payloads with acquired targets. A tag is simply a unique positive integer assigned to each Blue and Red combat agent for identification purposes (see section 1.2.2). The following pseudocode describes the algorithm used by `combat_ga` to ensure that payloads are delivered to targeted agents:

```
IF a fired round has reached a targeted cell THEN
    delta_x = agent x position – target x position
                when agent x position > target x position
        = target x position – agent x position otherwise
    delta_y = agent y position – target y position
                when agent y position > target y position
        = target y position – agent y position otherwise
    IF agent tag = target tag AND delta_x + delta_y ≤ 2 THEN
        IF current agent armor strength < round fire strength THEN
            engaged agent is killed
        ELSE
            engaged agent's new armor strength =
                current agent armor strength – round fire strength
        END
    END
END.
```

Thus, the fired round essentially searches a symmetric diamond-shaped CA neighborhood centered on the target cell and then delivers the fire to an agent whose tag matches the target tag encoded into the round.

3.1.3 Genomic Operational Fitness

In Figure 11, the data filter is shown to dynamically calculate the operational fitness of each opposing combat agent unit within a `combat_ga` simulation (and then either directly output the time series fitness data or route it into the GA). The "fitness" of a combat unit is an operational measure of its collective ability to successfully execute combat actions and is analogous to one or more combined combat unit measures of performance. Fitness is thus a metric that can be used to evaluate the operational effectiveness of a specific genomic configuration vector encoding of a combat unit's hardware capabilities and behaviors. To carry this out, a set of genome fitness functions or dynamic measures of performance was designed. These functions are evaluated at the end of each simulation time cycle update of the CA engine and represent the collective performance of all agents within a combat unit. The linear superposition of all fitness function values at the n^{th} time cycle t_n is called the survival fitness.

$$F_{surv}(t_n) = W_1F_1(t_n) + W_2F_2(t_n) + W_3F_3(t_n) + \ldots + W_NF_N(t_n), \qquad (10)$$

where $F_1(t_n)$, $F_2(t_n)$, $F_3(t_n)$, …, $F_N(t_n)$ are the values of the N fitness functions at t_n, W_1, W_2, W_3, …, W_N are fitness function weights (where $\Sigma_i\, W_i = 1$), and $0 \leq F_i(t_n) \leq 1$ for all functions. The survival fitness (so-named for its representation of perpetuated unit performance under stress) thus provides an objective function that can be optimized by the GA.

Figure 14 illustrates the process that the data filter uses to evaluate *Fsurv(tn)* across a Blue/Red combat simulation time window. For a combat simulation unfolding over M discrete time cycles, values of *Fsurv(tn)* are sampled at the end of each time cycle and then summed and normalized to the time window length tM. This results in the time-averaged survival fitness $\langle F_{surv} \rangle$, where $0 \leq \langle F_{surv} \rangle \leq 1$. It is this last measure that finally defines the relative fitness of both Blue and Red unit genome candidates within a multigenerational coevolutionary simulation. By constructing a multidimensional genomic hypersurface (representing combat interactions between different Blue and Red genome candidates), where each point on the hypersurface is weighted by Blue and Red values of $\langle F_{surv} \rangle$ associated with the corresponding combat interaction, the GA has a "fitness landscape" which it can navigate in its search to cooptimize both Blue and Red unit operational fitness.

In the following subsections, the individual fitness functions making up $F_{surv}(t_n)$ are discussed in detail. In these fitness functions, the terms "friendly" and "enemy" are used to distinguish between combat agents within the same unit vs. agents within an adversarial unit, respectively.

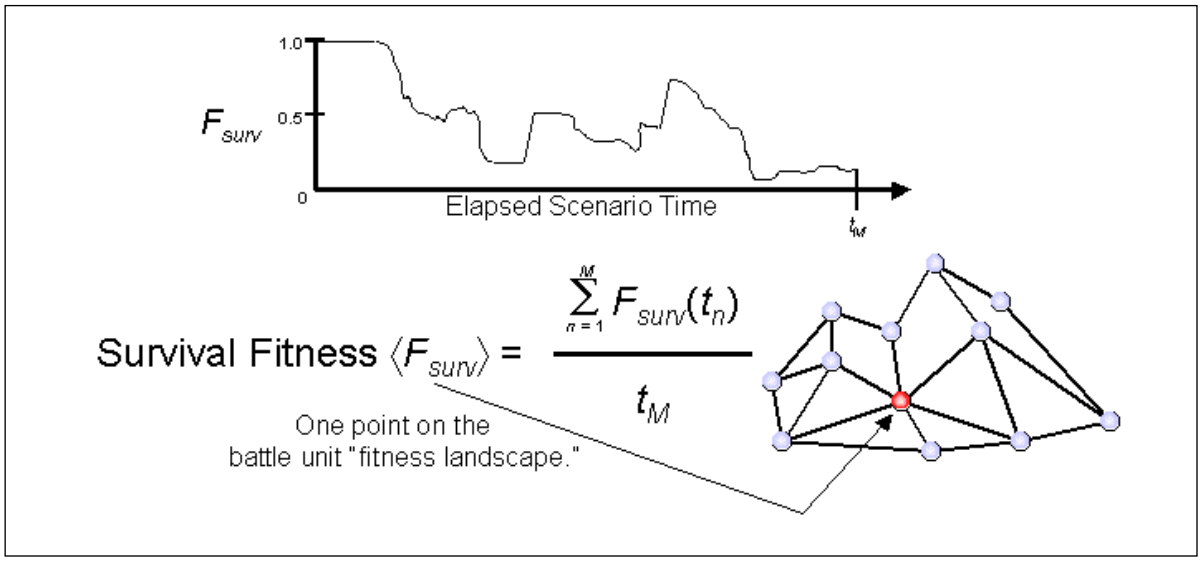


Figure 14. Evaluation of collective combat unit time-averaged survival fitness $\langle F_{surv} \rangle$.

Thus, the terms are relative to a specific unit and can be used in the context of either Blue or Red combat agents (e.g., the Blue unit is the "enemy" of the Red unit, while Red agents are all members of the same "friendly" unit).

3.1.3.1 $F_1(t_n)$*:* Prevent Enemy Penetration Into Friendly Territory

The first fitness function measures the degree to which a unit of defensive combat agents prevents enemy agents from penetrating into the defender's battlefield territory throughout the course of a simulation. The function is expressed as

$$F_1(t_n) = 1 - \frac{\sum_{\text{friendly cells}} \text{importance}_{\text{friendly cell}} * k_{\text{enemies in cell}}(t_n)}{\text{importance}_{\max} * k_{\text{initial alive enemies}}}, \quad (11)$$

where $\text{importance}_{\text{friendly cell}}$ is the territorial importance of a defender-owned lattice cell (quantified as a number between 0 and 128), $k_{\text{enemies in cell}}(t_n)$ is the number of enemy agents in the defender-owned lattice cell at the n^{th} simulation time cycle, $\text{importance}_{\max}$ is the maximum possible territorial importance of a cell (where $\text{importance}_{\max} > 0$), and $k_{\text{initial alive enemies}}$ is the total number of enemy agents within the simulation at the start of a simulation instance (where $k_{\text{initial alive enemies}} > 0$). The numerator of $F_1(t_n)$ is calculated by summing values of the product $\text{importance}_{\text{friendly cell}} * k_{\text{enemies in cell}}(t_n)$ over all defender-owned cells within the CA lattice. This fitness function utilizes the territorial importance gradient introduced in section 2.3.10.2 and ranges from a value of 1 (when all defender territory is clear of invasive enemy agents) to 0 (when all enemy agents are still alive and clustered at the most "important" cell within the defender unit's territory).

3.1.3.2 $F_2(t_n)$: Penetrate Into Enemy Territory

The second fitness function measures the degree to which a unit of invasive combat agents successfully penetrates into an enemy's battlefield territory throughout the course of a simulation. The function is expressed as

$$F_2(t_n) = \frac{\sum_{\text{enemy cells}} \text{importance}_{\text{enemy cell}} * k_{\text{friends in cell}}(t_n)}{\text{importance}_{\max} * k_{\text{initial alive friends}}}, \tag{12}$$

where $\text{importance}_{\text{enemy cell}}$ is the territorial importance of an enemy-owned lattice cell (quantified as a number between 0 and 128), $k_{\text{friends in cell}}(t_n)$ is the number of friendly invasive agents in the enemy-owned lattice cell at the n^{th} simulation time cycle, $\text{importance}_{\max}$ is the maximum possible territorial importance of a cell (where again $\text{importance}_{\max} > 0$), and $k_{\text{initial alive friends}}$ is the total number of friendly invasive agents within the simulation at the start of a simulation instance (where $k_{\text{initial alive friends}} > 0$). Similar to $F_1(t_n)$, the numerator of $F_2(t_n)$ is calculated by summing values of the product $\text{importance}_{\text{enemy cell}} * k_{\text{friends in cell}}(t_n)$ over all enemy-owned cells within the CA lattice. This fitness function also utilizes the territorial importance gradient and ranges from a value of 0 (when all enemy territory is clear of invasive friendly agents) to 1 (when all friendly agents are still alive and clustered at the most "important" cell within the enemy unit's territory).

3.1.3.3 $F_3(t_n)$: Friendly Combat Unit Survival

The third fitness function measures the fraction of friendly combat agents within an allied unit that remain alive during the course of a simulation. The function is expressed as

$$F_3(t_n) = \frac{k_{\text{alive friends}}(t_n)}{k_{\text{initial alive friends}}}, \tag{13}$$

where $k_{\text{alive friends}}(t_n)$ is the number of friendly agents within an allied unit that remain alive at the n^{th} simulation time cycle and $k_{\text{initial alive friends}}$ is the same as was defined in equation 9 (where again $k_{\text{initial alive friends}} > 0$). It should be noted that all agents have the same weighting relative to $k_{\text{alive friends}}(t_n)$ (i.e., a value of 1 given that the agent is alive) irregardless of the specific capabilities or combat role that any particular agent might have.

3.1.3.4 $F_4(t_n)$: Enemy Combat Unit Casualties

The fourth fitness function measures the fraction of enemy combat agents within an adversarial unit that are killed during the course of a simulation. The function is expressed as

$$F_4(t_n) = \frac{k_{\text{dead enemies}}(t_n)}{k_{\text{initial alive enemies}}}, \tag{14}$$

where $k_{\text{dead enemies}}(t_n)$ is the number of enemy agents within an adversarial unit that have been killed by the n^{th} simulation time cycle and $k_{\text{initial alive enemies}}$ is the same as was defined in equation

11 (where again $k_{\text{initial alive enemies}} > 0$). As is the case with $F_3(t_n)$, all agents contributing to $F_4(t_n)$ have the same weighting relative to $k_{\text{dead enemies}}(t_n)$ (i.e., a value of 1 given that the agent has been killed) irregardless of the specific capabilities or combat role that any particular agent might have.

3.1.3.5 $F_5(t_n)$: Sense the Battlefield

The fifth fitness function measures the fraction of the total CA universe collectively sensed by all sensor-equipped agents (in combination with stand-alone UGS units when available) within a combat unit during the course of a simulation. The function is expressed as

$$F_5(t_n) = \frac{k_{\text{sensed cells}}(t_n)}{k_{\text{total cells}}}, \tag{15}$$

where $k_{\text{sensed cells}}(t_n)$ is the number of lattice cells within the CA universe collectively sensed by a combat unit during the n^{th} simulation time cycle and $k_{\text{total cells}}$ is the total number of cells within the CA universe. It should be noted that this fitness function only measures whether a cell has been sensed and not the degree to which the sensed information is shared amongst agents within the combat unit.

3.1.3.6 $F_6(t_n)$: Jam Enemy Communications

The sixth fitness function measures the fraction of combat agents within a unit whose communication reception capability has been jammed by agents within an opposing unit during the course of a simulation. The function is expressed as

$$F_6(t_n) = \frac{k_{\text{jammed enemies}}(t_n)}{k_{\text{alive enemies}}(t_n)}, \tag{16}$$

where $k_{\text{jammed enemies}}(t_n)$ is the number of enemy agents whose communication reception has been jammed by an opposing unit of friendly agents during the n^{th} simulation time cycle and $k_{\text{alive enemies}}(t_n)$ is the total number of surviving enemy agents evaluated at the same time cycle. In the case where $k_{\text{alive enemies}}(t_n) = 0$, $F_6(t_n)$ is set equal to 1, it should be noted that communications jamming is accomplished either by jammer-equipped combat agents or by stand-alone jammer bombs (when available for deployment).

3.1.3.7 $F_7(t_n)$: Prevent Enemy Penetration Into Friendly Territory With Dynamic Renormalization

The seventh fitness function is identical to $F_1(t_n)$ (i.e., prevent enemy penetration into friendly territory) except that the denominator in the second term is now adaptively renormalized to the population of *surviving* enemies at a simulation time cycle rather than the initial enemy agent population. The function is expressed as

$$F_7(t_n) = 1 - \frac{\sum_{friendly\ cells} \text{importance}_{\text{friendly cell}} * k_{\text{enemies in cell}}(t_n)}{\text{importance}_{\max} * k_{\text{alive enemies}}(t_n)}, \quad (17)$$

where all variable definitions are the same as in equation 11 except for $k_{\text{alive enemies}}(t_n)$, which is the same as defined in equation 16. As with $F_6(t_n)$, $F_7(t_n)$ is set equal to 1 when $k_{\text{alive enemies}}(t_n) = 0$. This dynamic renormalization provides evolutionary pressure that forces the defensive combat unit to prevent *any* degree of enemy unit penetration into friendly territory throughout the duration of a simulation instance (i.e., friendly unit fitness is minimized as long as there is at least one surviving enemy agent *and* all surviving enemies penetrate into the defensive unit's most important territory).

3.1.3.8 $F_8(t_n)$: Penetrate Into Enemy Territory With Dynamic Renormalization

The eighth fitness function is identical to $F_2(t_n)$ (i.e., penetrate into enemy territory) except that the denominator in the second term is now adaptively renormalized to the population of *surviving* friends at a simulation time cycle rather than the initial friendly agent population. The function is expressed as

$$F_8(t_n) = \frac{\sum_{enemy\ cells} \text{importance}_{\text{enemy cell}} * k_{\text{friends in cell}}(t_n)}{\text{importance}_{\max} * k_{\text{alive friends}}(t_n)}, \quad (18)$$

where all variable definitions are the same as in equation 12 except for $k_{\text{alive friends}}(t_n)$, which is the same as defined in equation 13. Here, $F_8(t_n)$ is set equal to 0 when $k_{\text{alive friends}}(t_n) = 0$. This dynamic renormalization provides evolutionary pressure that forces the invasive combat unit to infiltrate enemy territory at almost *any* casualty cost to the invaders (i.e., friendly unit fitness is maximized as long as there is at least one surviving friendly agent *and* all surviving friends penetrate into the enemy's most important territory).

3.1.3.9 $F_9(t_n)$: Collective Friendly Situational Awareness of Friends

The ninth fitness function measures the fraction of surviving combat agents within a unit whose current situational status (i.e., location, identity, and capability status) is known by all other agents within the unit (due to network-centric sensor data sharing via the unit communication network). The function is expressed as

$$F_9(t_n) = \frac{\sum_{\text{all cells}} m_{\text{friendly SA messages concerning friends}}(t_n)}{\left[k_{\text{alive friends}}(t_n)\right]^2}, \quad (19)$$

where $m_{\text{friendly SA messages concerning friends}}(t_n)$ is the number of SA data messages residing in a cell, which were broadcast by friendly agents reporting on other sensed friends within the combat unit at the n^{th} simulation time cycle, and $k_{\text{alive friends}}(t_n)$ is the same as defined in equation 13. As with

$F_8(t_n)$, $F_9(t_n)$ is set equal to 0 when $k_{\text{alive friends}}(t_n) = 0$. Since nonpropagating SA messages are associated with combat agents and each agent has at most one copy of an SA message reporting on a specific friendly agent, dividing the total number of friendly agent reports (collected throughout all cells within the CA universe) by the square of the number of friendly agents measures the degree of collective friendly situational awareness within a combat unit.

3.1.3.10 $F_{10}(t_n)$: Collective Friendly Situational Awareness of the Enemy

The 10th and final fitness function measures the fraction of surviving combat agents within an enemy unit whose current situational status (i.e., location, identity, and capability status) is known by all agents within the opposing friendly unit (due to network-centric sensor data sharing via the friendly unit communication network). The function is expressed as

$$F_{10}(t_n) = \frac{\sum_{\text{all cells}} m_{\text{friendly SA messages concerning enemies}}(t_n)}{k_{\text{alive friends}}(t_n)\, k_{\text{alive enemies}}(t_n)}, \tag{20}$$

where $m_{\text{friendly SA messages concerning enemies}}(t_n)$ is the number of SA data messages residing in a cell, which were broadcast by friendly agents reporting on sensed enemies within the opposing combat unit at the n^{th} simulation time cycle, and $k_{\text{alive friends}}(t_n)$ and $k_{\text{alive enemies}}(t_n)$ are the same as defined in equations 13 and 16, respectively. Here, $F_{10}(t_n)$ is set equal to 0 when $k_{\text{alive friends}}(t_n) = 0$ and is set equal to 1 when $k_{\text{alive enemies}}(t_n) = 0$. Similar to the case with $F_9(t_n)$ regarding friendly agent awareness, dividing the total number of friendly agent reports regarding enemy agents (again collected throughout all cells within the CA universe) by the product of the number of surviving friendly and enemy agents allows $F_{10}(t_n)$ to measure the degree of collective friendly unit situational awareness of the enemy.

3.1.4 Genetic Operations

As described in section 3.1.1, the GA instantiated within the `reproduce` routine acts to generate new combat unit configuration genome candidates by operating upon the fittest "parent" genomes (based on the preselected fitness functions introduced in the previous section). There are two different types of genetic operators built into the GA code; they are as follows:

- crossover - a process that exchanges sets of contingent chromosome vectors between two parent genomes, and
- point mutation - a process that randomly changes the allele values of individual genes within a chromosome.

An example illustrating the application of these genetic operators to two parent genomes is depicted in Figure 15. In this example, a point within the crossover variance is selected within each parent chromosome. The crossover variance indicates the preset positional range within a genome measured outward from the center point where the genome can be separated into two

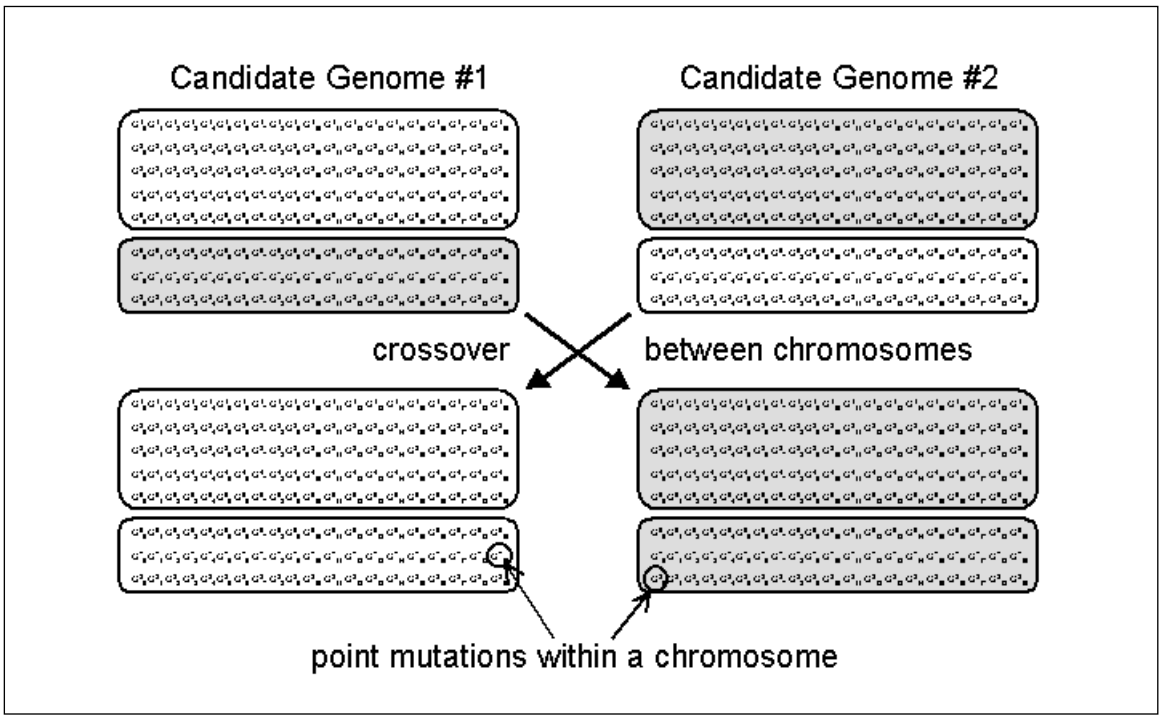


Figure 15. Genetic operations on two candidate combat unit genomes.

sets of constituent chromosomes.* These chromosome sets (indicated in the figure by the white and gray shadings) are exchanged between candidate genomes nos. 1 and 2, resulting in two new genomes made up of the same chromosome "primitives" that form the bodies of the parent genomes. Then, point mutations are executed on randomly-selected genes within each chromosome, which essentially acts to create new types or "species" of combat agents. It should be noted that the crossover operator treats individual chromosome vectors as indivisible so that chromosomes within a parent genome remain intact after crossover. Changes within a chromosome can thus *only* occur via point mutation, which acts to ensure perpetuated diversity of battlefield capabilities among the agents making up a combat unit (Ilachinski 1996).

3.1.5 Genome Cost Filtering

The final process still to be addressed within the Blue/Red coevolutionary cycle described in section 3.1.1 is cost-constrained evolution, where each "hardware-based" gene within all agent chromosomes in a genome (i.e., chromosome genes G_0 through G_{18} inclusive) is allowed to mutate within a preset cost-constrained interval. In order to incorporate cost-constrained evolution into the GA, the cost-filtering routine was introduced into the coevolution software framework (see Figure 11). This routine works in conjunction with both the `random_genome` and `reproduce` routines and a shared GA configuration file to filter each newly-created candidate genome based on its compliance with a set of cost constraints. Genomes that pass the

* Although both parent genomes must share the same crossover variance, the genome separation point (which must lie within the crossover variance range) need not be the same for both parent genomes.

cost filter compliance test are directed into the Blue and Red genome candidate populations as appropriate, while failure to comply causes the genome candidate to be rejected. The cost constraints making up the cost filter are discussed next.

The cost filter assumes that the total cost of a genome is simply

$$cost_{genome} = \sum_{chromosomes} \sum_{gene=G_0}^{G_{18}} cost_{gene}, \tag{21}$$

where $cost_{gene}$ is the implementation cost (in generic economic cost units) of a particular hardware gene. The implementation cost function used to calculate $cost_{gene}$ is illustrated in Figure 16. Here, the implementation cost of a gene is characterized by the following two parameters: (1) the installation cost $cost_{gene}^{install}$, which is the cost required to initially "install" or turn on the hardware gene, and (2) the incremental cost $cost_{gene}^{increment}$, which is the cost required to increase the gene allele value by one unit. The final gene cost is then

$$cost_{\text{gene}} = cost_{gene}^{install} + \left(\text{k}_{\text{allele}} - 1\right) * cost_{gene}^{increment}, \tag{22}$$

where k_{allele} is the integer value of the gene allele. Then, the set of cost constraints making up the genome cost filter includes (a) $cost_{chromosome}^{max}$ (the maximally allowed total cost of an individual chromosome) and (b) $cost_{genome}^{max}$ (the maximally allowed cost of a genome). Upon generation by either the `random_genome` or `reproduce` routines, constraint (b) is first applied to a genome candidate after its cost is calculated via equations 21 and 22. Given that the genome candidate cost $\leq cost_{genome}^{max}$, constraint (a) is next applied. If the genome candidate cost $\leq cost_{chromosome}^{max}$, then the candidate is added to the Blue or Red genome pool as appropriate.

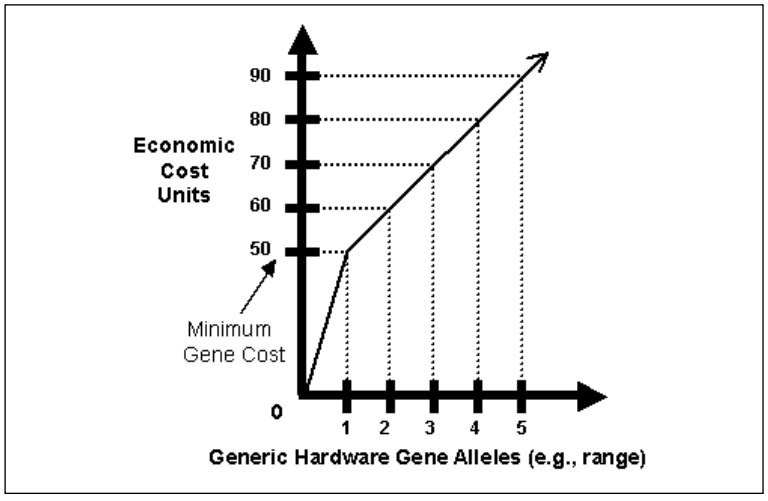


Figure 16. GA cost function for hardware-specific genes within a chromosome.

3.1.6 Information Operations Simulations

In the following subsections, the combat scenario, GA configuration conditions, and simulation outputs for IO simulations utilizing the coevolutionary software framework and constituent software elements described in sections 3.1.1–3.1.5 are presented and discussed.

3.1.6.1 Combat Scenario

The combat scenario depicted in Figures 17 and 18 was used in conjunction with the coevolutionary software framework depicted in Figure 11 to run GA benchmark tests. Figure 17 illustrates the flat virtual battlefield terrain (generated by running the `terrain` routine with the "empty" option selected) occupied by Blue and Red combat agents, which begin a simulation run in the southwest and northeast quadrants of the terrain, respectively. Combat agents are initially positioned at the start of a simulation instance in randomly selected cells within adjustable subquadrants of the CA universe by running the `place` routine with the "corners" option selected.

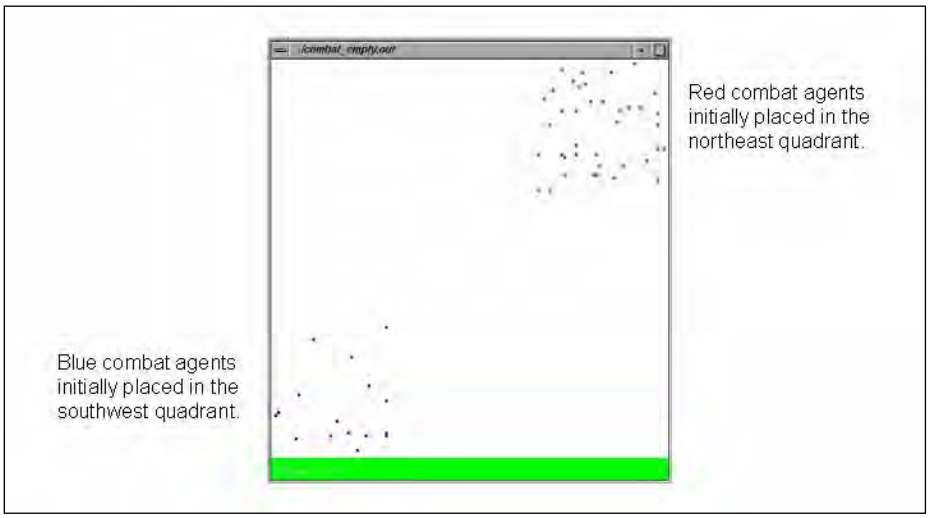


Figure 17. Basic Blue/Red combat scenario set on a flat plane.

Figure 18 illustrates the territorial "importance" gradient that is superimposed over terrain and is used to motivate combat agent movement towards a goal (as previously discussed in section 2.3.10.2). Within this gradient, blue-colored terrain is "neutral" and thus minimally important, while red-colored terrain represents operationally "critical" territory. Thus, combat agent territory unfolds from the "no-man's-land" along the center diagonal towards critical Blue- and Red-controlled objective points in the southwest and northeast corners of the 2-D battlespace, respectively. Based on the allele values assigned to genes G_{25} through G_{28} within a combat agent chromosome, an agent will be driven to either advance or retreat along successive importance gradient contours.

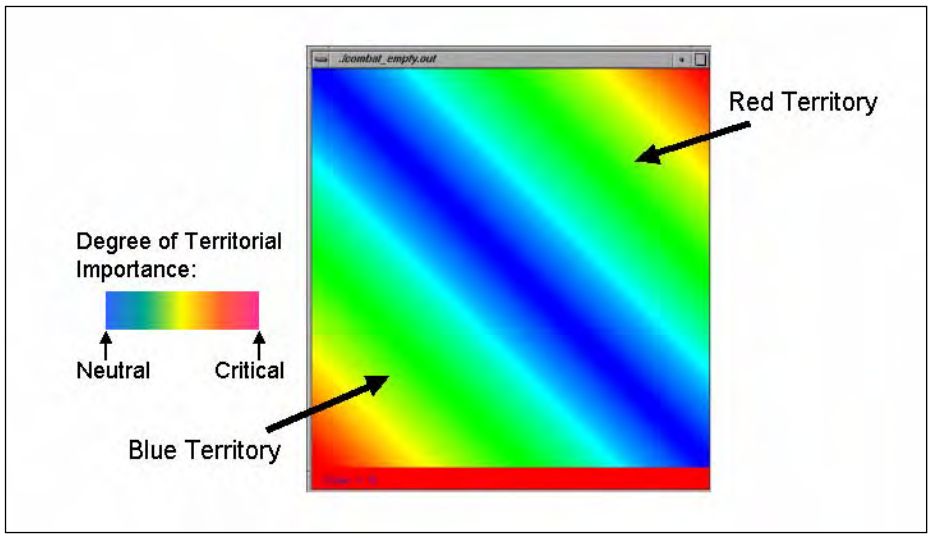


Figure 18. Territorial importance gradient associated with the basic Blue/Red combat scenario.

3.1.6.2 Coevolutionary Simulation Configuration

Once the combat scenario has been properly delineated, the GA, survival fitness function, and Blue/Red genome configuration files are defined. These configuration files are presented in Tables 1–3, respectively. The parameters listed in Table 1 are used by the GA to control the coevolutionary dynamics of both the Blue and Red candidate genome populations. The second parameter (maximal number of allowed successive generations) provides an automated termination point for a coevolutionary simulation; the user can, however, manually terminate a simulation prior to this point if so desired.

Next, the parameters listed in Table 2 are used to set the duration of a Blue/Red combat instance and to construct the respective survival fitness objective functions for both Blue and Red combat units. Here, Blue's primary mission objective is to prevent *any* level of surviving enemy penetration into friendly territory (characterized by the fitness function $F_7[t_n]$) with a secondary objective to invade the enemy's territory no matter what the friendly attrition cost (characterized by the fitness function $F_8[t_n]$), while the reverse is true for Red. These survival fitness objective functions force both combat units to search for capability configurations and tactical behaviors that serve to achieve a multiobjective goal.

Finally, the parameters listed in Table 3 define the size, cost, and gene allele value constraints that are imposed upon both Blue and Red genome candidates throughout the entirety of a multigenerational coevolutionary simulation run. The first constraint was chosen to imply that Blue was logistically constrained to maintaining a small unit, while Red was not. The net genome cost constraint was chosen to imply that Blue, although logistically constrained, had

Table 1. The GA configuration file used in the coevolutionary simulation.

Parameter Description	Value
Size of the CA lattice.	136 cells × 136 cells
Maximal number of successive generations allowed within a coevolutionary simulation.	2400 generations
The size of both the Blue and Red candidate genome pools that should be maintained throughout a coevolutionary simulation.	30 genomes
The number of fittest genomes to use as parents for genetic breeding of new genome candidates.	5 genomes
The number of simulation instances to run for each Blue/Red genome combat pairing.	3 instances
The crossover variance for both Blue and Red parent genomes used in genetic breeding.	0.30
The probability that a particular chromosome within a genome is selected for gene mutation.	0.20
The probability that a particular gene within a chromosome will undergo mutation given that the chromosome has been selected for this operation.	0.01

Table 2. The survival fitness configuration file used in the coevolutionary simulation.

Parameter Description	Value	
Total simulation run time.	300 time cycles	
$F_{surv}(t_n)$ fitness function weights.	Blue genome	Red genome
$F_1(t_n)$: prevent enemy penetration into friendly territory.	0	0
$F_2(t_n)$: penetrate into enemy territory.	0	0
$F_3(t_n)$: friendly combat unit survival.	0	0
$F_4(t_n)$: enemy combat unit casualties.	0	0
$F_5(t_n)$: sense the battlefield.	0	0
$F_6(t_n)$: jam enemy communications.	0	0
$F_7(t_n)$: prevent enemy penetration into friendly territory with dynamic renormalization.	0.7	0.3
$F_8(t_n)$: penetrate into enemy territory with dynamic renormalization.	0.3	0.7
$F_9(t_n)$: collective friendly situational awareness of friends.	0	0
$F_{10}(t_n)$: collective friendly situational awareness of the enemy.	0	0

greater economic resources at its disposal than did Red. The allele value constraints inform the GA of maximal-allowed values of specific mutable genes within a chromosome (i.e., speed, sensor range, fire range, fire strength, armor strength, carried fire rounds, carried land mines, and carried RF jammer bombs), as well as the intentionally fixed allele values assigned to seven immutable genes (i.e., sensor detection probability, sensor false alarm probability, carried UGS

Table 3. The genome configuration file used in the coevolutionary simulation.

Parameter Description	Blue Genome	Red Genome
Maximum allowable number of chromosomes within a genome.	25	1024
Maximum allowable net genome cost.	800,000 cost units	500,000 cost units
Maximum allowable combat agent fire range (G_0).	10 cells	10 cells
Maximum allowable combat agent fire strength (G_2).	20	20
Maximum allowable number of carried firepower rounds (G_3).	255	255
Maximum allowable combat agent armor strength (G_4).	20	20
Maximum allowable combat agent sensor range (G_5).	40 cells	20 cells
Maximum allowable number of carried land mines (G_{11}).	255	255
Maximum allowable number of carried RF jammer bombs (G_{12}).	255	255
Maximum allowable combat agent speed (G_{17}).	10	10
Fixed sensor probability of target detection (G_6).	0.95	0.95
Fixed sensor probability of false alarm (G_7).	0.05	0.05
Fixed combat agent communication range (G_8).	40 cells	20 cells
Fixed combat agent RF jamming range (G_9).	5 cells	5 cells
Fixed number of carried UGS units (G_{10}).	0	0
Fixed UGS unit deployment range (G_{13}).	0	0
Fixed probability of retaliating against an enemy (G_{19}).	1.00	1.00

units, UGS deployment range, communication and RF jamming ranges, and retaliation probability).* Chromosome genes that are not specifically addressed in this figure are thus allowed to mutate without allele constraints (except for preconstrained genes such as probabilities and behavioral weights). The allele value constraints were chosen to imply that Blue also had access to superior sensor and communication hardware than did Red.

3.1.6.3 Simulation Results

A coevolutionary Blue-vs.-Red simulation (using the scenario and GA/fitness/genome configurations described in sections 3.1.6.1 and 3.1.6.2, respectively) was run on a 32-CPU SGI Origin 3800 computer. Here, Blue/Red combat instances for each evolutionary generation of candidate genomes were run in parallel, after which calculated survival fitness values were centrally averaged and reported. Three combat simulation instances were executed (and then averaged) for each Blue/Red genome candidate pairing with 30 pairings per generation. The latter situation was iteratively realized by randomly selecting one candidate genome from both Blue and Red genome pools for a combat pairing and then removing the selected genomes from their respective pools and repeating the random selection process until all genomes have been

* These seven genes were predefined as immutable, implying a scenario where sensor, communication, and RF jamming hardware capabilities were constrained by currently available commercial off-the-shelf (COTS) technology (which is not assumed to include UGS technology) and an agent will always return fire when engaged.

paired for combat. Each combat simulation was run for 300 time cycles (see Table 2), at which point values of the time-averaged survival fitness $\langle F_{surv} \rangle$ for both Blue and Red genomes were evaluated and reported. Then, for both Blue and Red units, the top five candidate genome scorers from each generation (see Table 1) were used to produce the next generation of 30 candidate genomes via random mating pairs.

Figure 19 depicts the coevolutionary dynamics of Blue/Red average and maximal time-averaged survival fitness levels (y-axis) measured across 1360 generations (x-axis), where Blue and Red values of $\langle F_{surv} \rangle$ are based on the fitness function weights presented in Table 2. Here, Red quickly "discovers" unit configurations that serve to maintain its average fitness within the interval [0.38, 0.43] which, in turn, motivates Blue to compensate with new configurations that likewise drive average Blue fitness to the interval [0.49, 0.59]. This coevolutionary process nicely illustrates the "Red Queen Principle" from evolutionary biology, where for an evolutionary system, continuing development is needed just in order to maintain its fitness level relative to the systems it is coevolving with (Van Valen 1973). Since, due to computational runtime constraints, the coevolutionary process was prematurely terminated before any strong evidence of survival fitness convergence was evidenced, it remains unclear whether the Blue and Red combat units would ever break out of the "Red Queen" oscillatory pattern illustrated in the figure. If the current theory of coevolutionary neutral mutations (where gene mutations produce insignificant variations in relative fitness [Bar-Yam 1997; Whitfield 2002]) is applicable within the current context, then fitness convergence might never be achievable without relaxing some of the gene allele and cost constraints defined in Table 3.

Further evidence of continuing development of both Blue and Red combat units throughout the coevolutionary process displayed in Figure 19 is provided by a classification analysis of Blue and Red genome populations. Table 4 defines a set of combat agent types or "classes" specified by agent sensor and firepower ranges, while Figure 20 illustrates the correlated coevolutionary dynamics of optimal Blue and Red unit configurations relative to this set of classes. Sensor classes include zero-, short-, medium-, and long-range sensor capability, while firepower classes are restricted to zero-, very short-, and short-range fire. The plots in Figure 20 measure, for the maximal-fitness genome in both populations, the number of Blue/Red combat agents within a particular sensor/firepower class (y-axis) across the 1360 generations making up the coevolutionary process (x-axis).[*] Here, Blue unit configuration is observed to oscillate between variable mixtures of medium- and long-range sensor agents with variable-range firepower weapon capability, finally settling into a mode where most agents have midrange sensor and short-range firepower capability. The Red unit, on the other hand, which must execute the more operationally challenging primary role of invader, coherently explores sensor/firepower classes

[*] A curve-smoothing algorithm has been applied to these plots (that averages class population levels over 10 successive generations) to make them easier to read.

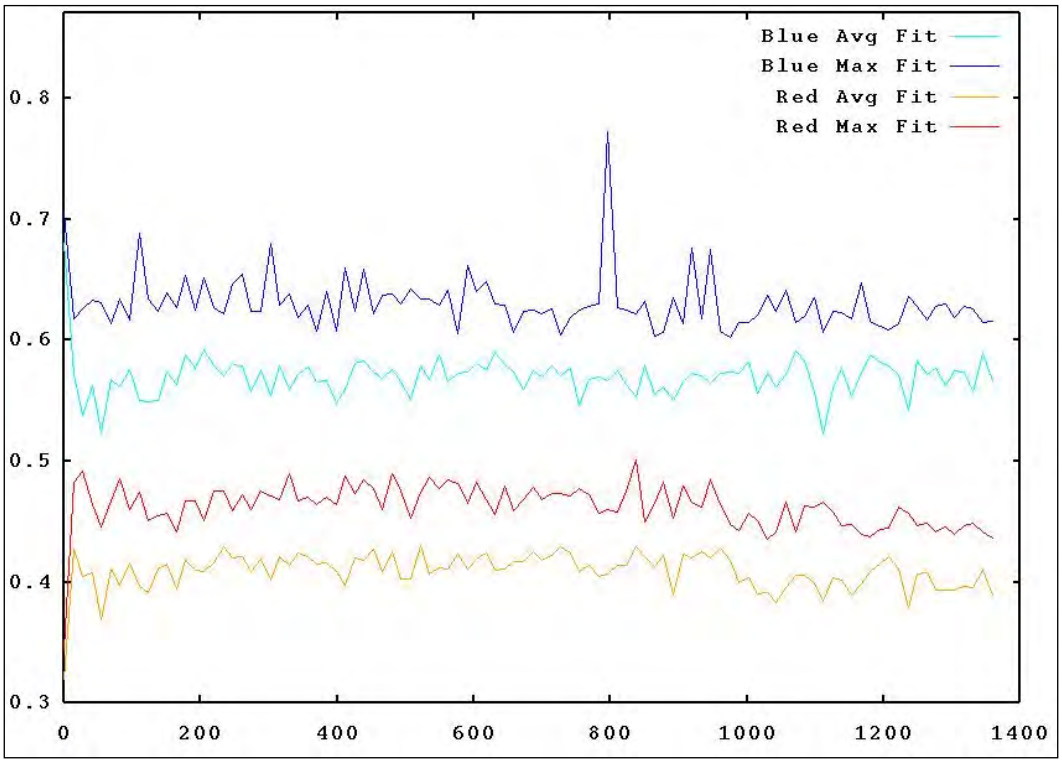


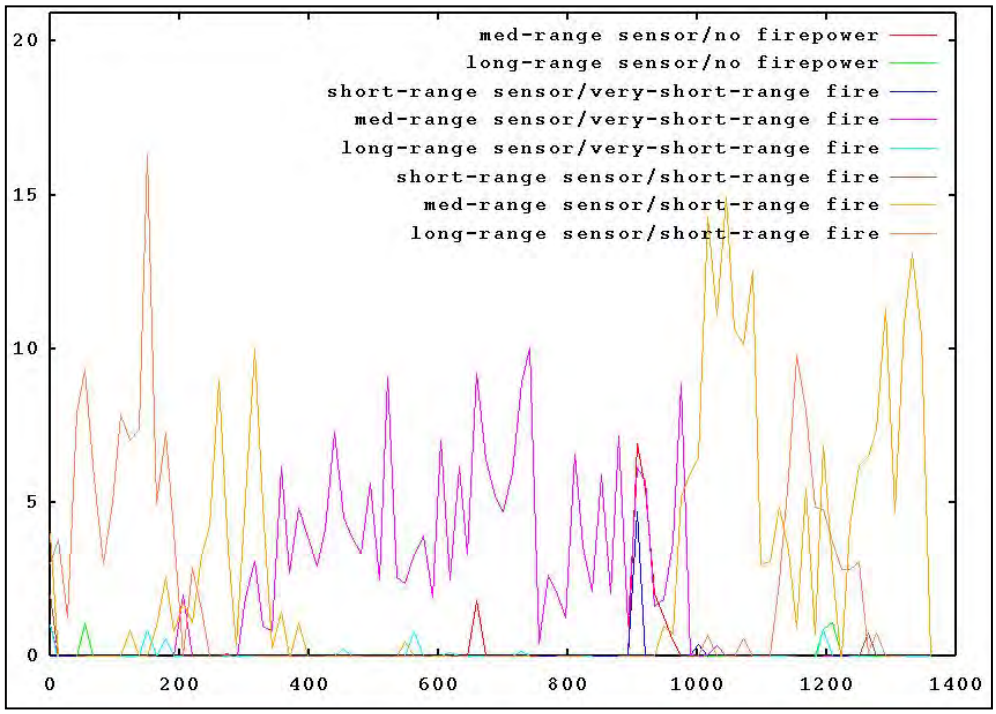
Figure 19. Coevolutionary dynamics of Blue/Red time-averaged survival fitness levels (y-axis) measured across 1360 coevolutionary generations (x-axis).

Table 4. Definition of a set of combat agent classes based on sensor/firepower range combinations.

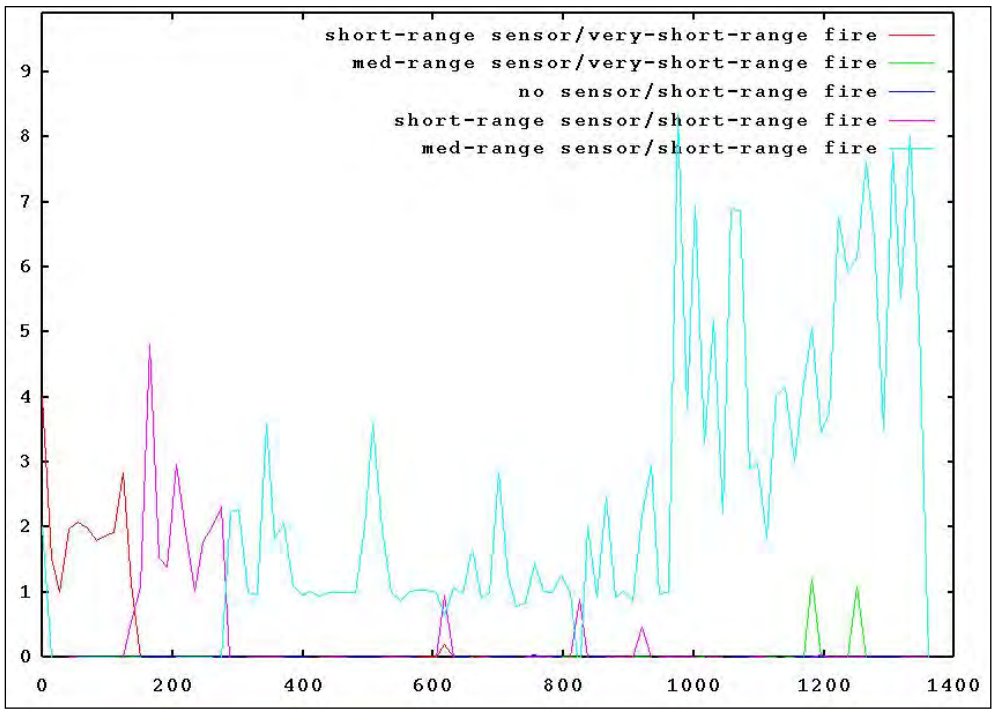
Combat Agent Class	Gene G_5 (Sensor Range)	Gene G_0 (Firepower Range)
No sensor/no firepower	0	0
Short-range sensor/no firepower	1–10 cells	0
Medium-range sensor/no firepower	11–25 cells	0
Long-range sensor/no firepower	26–40 cells	0
No sensor/very short-range firepower	0	1–5 cells
Short-range sensor/very short-range firepower	1–10 cells	1–5 cells
Medium-range sensor/very short-range firepower	11–25 cells	1–5 cells
Long-range sensor/very short range firepower	26–40 cells	1–5 cells
No sensor/short-range firepower	0	6–10 cells
Short-range sensor/short-range firepower	1–10 cells	6–10 cells
Medium-range sensor/short-range firepower	11–25 cells	6–10 cells
Long-range sensor/short-range firepower	26–40 cells	6–10 cells

until finally settling on a medium-range sensor and short-range fire for most agents. It should be noted that total unit populations are variable across successive generations due to genome size and cost filtering.

Next, Table 5 defines a second set of combat agent classes specified by agent behavioral weighting combinations, while Figure 21 illustrates the correlated coevolutionary dynamics of optimal Blue and Red unit configurations relative to this new set of classes. Behavioral classes



(a)



(b)

Figure 20. Coevolutionary dynamics of optimal Blue and Red unit configurations relative to the set of combat agent sensor/firepower range classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

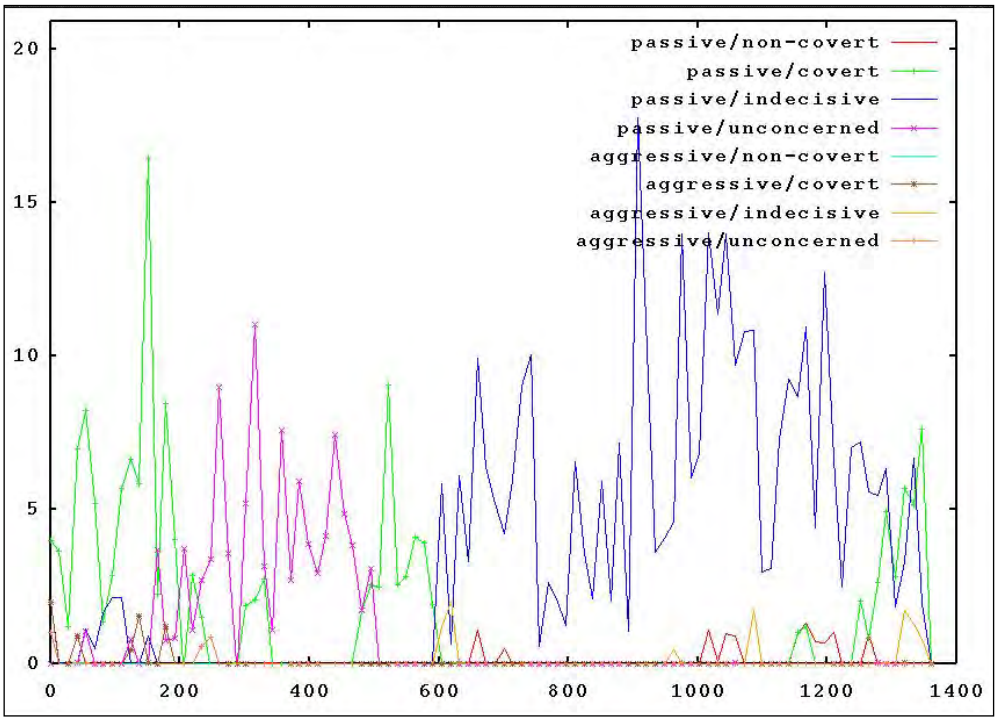
Table 5. Definition of a set of combat agent classes based on behavioral gene weighting combinations.

Combat Agent Class	Gene G_{20} (Prob. of Initiating Attack)	Gene G_{23} (Attraction Towards Enemy CM)	Gene G_{24} (Repulsion From Enemy CM)
Passive/noncovert	≤ 0.50	> 0.50	≤ 0.50
Passive/covert	≤ 0.50	≤ 0.50	> 0.50
Passive/indecisive	≤ 0.50	> 0.50	> 0.50
Passive/unconcerned	≤ 0.50	≤ 0.50	≤ 0.50
Aggressive/noncovert	> 0.50	> 0.50	≤ 0.50
Aggressive/covert	> 0.50	≤ 0.50	> 0.50
Aggressive/indecisive	> 0.50	> 0.50	> 0.50
Aggressive/unconcerned	> 0.50	≤ 0.50	≤ 0.50

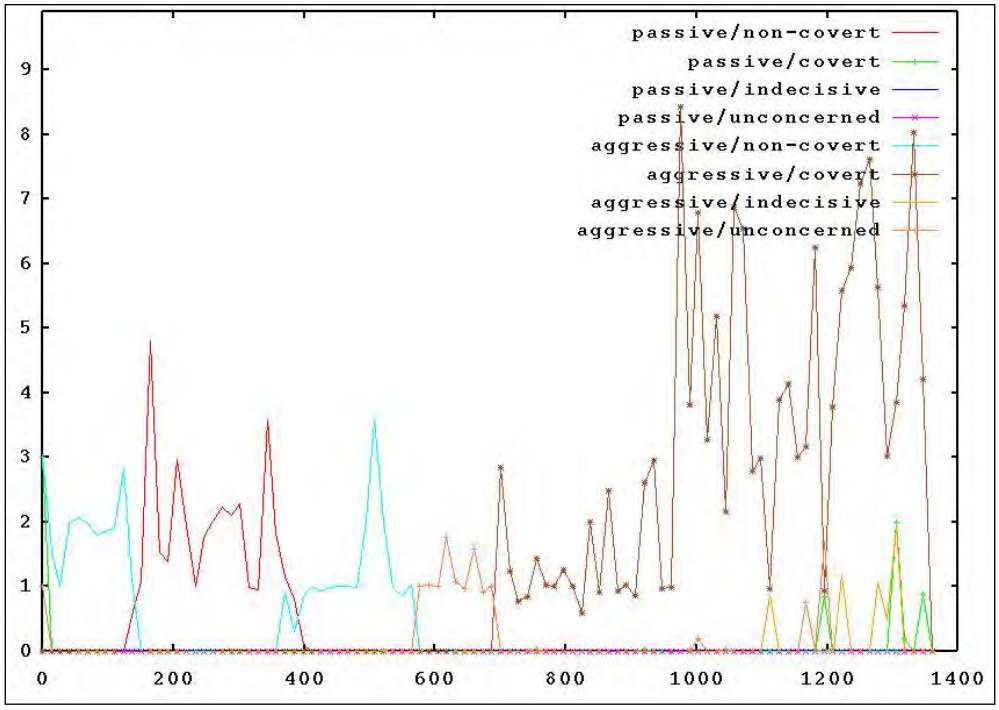
are formed by combining binary gene values (i.e., allele levels that are either ≤ 0.50 or > 0.50) of agent attack initiation probability and attraction towards/repulsion from enemy unit CM. Similar to Figure 20, the plots in Figure 21 measure (again for the maximal-fitness genome in both populations) the number of Blue/Red combat agents within a particular behavioral class across the coevolutionary process. In this case, Blue seems to haphazardly explore various classes, then settling on a passive/indecisive behavioral mode for over 600 generations, and finally switching over to a mixture of passive/indecisive and aggressive/indecisive agents at the end of the coevolutionary run. This is not surprising, in that Blue's primary mission is defensive and thus Blue agents can afford to be "indecisive" (effectively immobile) and less "aggressive" than Red agents. After an initial exploratory period, Red, on the other hand, first "discovers" a behavioral configuration where all agents act in an aggressive (firing a weapon without provocation) and noncovert (moving directly towards Blue agents) manner that serves to maintain its operational fitness. Then, when Blue counters with behaviors that stress Red fitness, Red, in turn, progressively adopts passive and noncovert, aggressive and noncovert, aggressive but indecisive (trying to move both towards/away from Blue), and aggressive but covert modes of behavior, finally settling on the last mode (with a small fraction of the Red unit continuing to explore other behaviors). Again, Red faces the greater evolutionary pressure since its primary mission is invasive in nature and thus benefits by coherently moving towards better strategic behavioral modes.

The preceding classification examples were presented to illustrate two limited perspectives on Blue/Red coevolutionary dynamics by focusing on specific genes within the combat agent chromosome. A true picture of the coevolutionary dynamics can only emerge by portraying the 20-dimensional response hypersurface generated by class population levels based on the 19 mutable genes within the agent chromosome (with genome generation providing the last degree of freedom).* While this is, of course, impossible to illustrate, it is feasible to provide 2-D "slices" of the response hypersurface depicting class population levels (as a function

* This also assumes that the behavioral attraction weights contained in genes G_{21}, G_{23}, G_{25}, and G_{27} can be combined with the repulsion weights contained in genes G_{22}, G_{24}, G_{26}, and G_{28}, respectively, thus reducing the number of dimensions required to track mutable genes from 23 down to 19.



(a)



(b)

Figure 21. Coevolutionary dynamics of optimal Blue and Red unit configurations relative to a set of combat agent behavioral classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

of gene-specific classes) across the multigenerational coevolutionary process. Plots of these slices are included in the Appendix.

Careful examination of these gene classification plots reveals, as was initially suggested by the plots in Figures 20 and 21, that Red's hardware capability evolution was generally more directed than Blue's. At the end of the 1360-generation coevolutionary simulation, both Blue and Red units end up with the maximal-permitted fire range (i.e., "short range" firepower), "high" fire strength, two crew members,* "fast" maximum speed, and air-travel capability.†

However, Blue firepower round supply moves up and then down, while Red supply steadily increases. Both Blue and Red armor strengths start "high" and then decrease (with Red starting to rise again at the end of the coevolutionary simulation). Blue sensor range continuously jumps up and down, while Red settles on a "moderate" sensor range and stays there. The values of Blue's land mine supply and mine/jammer bomb P_{deploy} also jump up and down, while Red again settles on a "moderate" level for both these genes well before the end of the simulation.

There are other specific asymmetric capability levels between Blue and Red that probably play a key role in facilitating Blue's superior survival fitness levels. Although Blue (principally the defender) ends up with a "low" firepower P_{hit} level while Red (principally the invader) finishes with a "high" value of that gene, Blue quickly settles on and stays with a "very large" jammer bomb supply while the poorer Red is forced to stay with a "moderate" supply level. This allows the defensive Blue unit greater capability to disrupt acquired target data sharing amongst agents in the invasive Red unit. Also, Blue's signature strength is maintained at a "low" level, while Red signature strength is forced to stay at a "high" level (due to lower levels of available financing). This asymmetry allows the Blue unit to generally avoid Red acquisition and fire (even though Red's capability to deliver a target hit is greater), while the Red unit's acquisition and subsequent engagement by Blue forces is relatively guaranteed by comparison.

The evolution of the behavioral modes within the Blue and Red units is a more complicated issue. Blue adopts and stays with a "moderate" likelihood of initiating an attack, while Red gradually settles on a "high" initiate attack probability; this makes sense given their respective defender and invader roles. However, although both Blue and Red end, up acting "indecisive" about moving towards/away from friendly agent clusters, Red quickly adopts a tactic to covertly avoid enemy agent clusters (while Blue remains indecisive regarding this tactic). While neither the Blue nor Red units are prone to stay on their own territory, in the end, Blue wants to invade Red's territory (even though the former's primary mission is defensive) while Red is again "indecisive" regarding enemy territory invasion (thus, only penetrating the outer reaches of Blue territory). Finally, both Blue and Red perpetuate a "low" probability of randomly changing

* There is a mutational bias in the GA so that an agent will essentially *never* have only one crew member (this was programmed to conform to the convention of functional redundancy in the human crew of a military platform).

† This mode of travel is likely a code artifact, since the flat terrain used in this simulation provides no operational advantage of using air flight over ground transport.

speed and direction, suggesting that occasional unpredictable movement on the part of an agent can help to maintain the survival fitness of its unit within a volatile environment. It is possible that the Red unit's generally inferior hardware capabilities prevent it from settling upon a straightforward set of behavioral tactics (i.e., the simultaneous attraction/repulsion behavior characteristic of indecision doesn't tend to facilitate achievement of a strategic goal). It is also very likely that the static nature of agent behavioral modes *during* a simulation instance doesn't provide the adaptive capability necessary for a unit to strategically exploit situations that evolve during the simulated combat.

It is interesting to try to characterize the degree of speciation within these results, where speciation refers (in the current context) to the process where agents within a combat unit evolve differentiated hardware capabilities and behavioral modes within the same evolutionary generation. The degree of agent speciation can be qualitatively assessed over the coevolutionary multigenerational simulation run via visual inspection of the classification plots in the Appendix. The difference in speciation between the Blue and Red combat units is immediately obvious. While neither Blue nor Red units consistently engage in the speciation process, Blue is considerably more active in this regard than Red. In fact, the Red unit appears to coherently avoid speciation across all chromosome genes except when adopting a new gene class (in which case, transient speciation occurs only as all agents within the unit progressively switch from one class to another). This is likely due to the mechanics of the GA (i.e., crossover *within* an agent chromosome is avoided) in combination with Red's more strategically complex role as the invader (where agent speciation must be combined with dynamic C2 in order to be operationally effective).

In the next section, the embedding of a quasiscripted combat agent C2 process directly into the CA simulation engine (and the subsequent IO simulation results) is explored in detail.

3.2 Enhanced Network-Centric Quasiscripted Simulation

The second and enhanced version of the CA simulation engine, "`combat_proto`" (so named for its utility in prototyping more complex combat agent behaviors) models the deployment and function of notional internetted UGS and multihop message relaying, and thus extends the network-centric capabilities of `combat_ga`. However, since `combat_proto` (also written in the Cellang programming language) employs a semiscripted time-dependent C2 process directly built into the associated code, this CA simulation engine cannot be used in conjunction with the GA (which assumes combat agents to maintain stationary behaviors throughout a combat simulation instance).

3.2.1 Simulation Framework

Since the simulation of UGS networks and multihop message relaying necessitates the extensive addition of new time steps into the CA update time cycle, `combat_proto` is ill-suited for use within a multigenerational coevolutionary simulation. In addition, the introduction of a

quasiscripted C2 process directly into the CA code overrides the static behavioral modes that characterize Blue/Red combat in the previously-discussed simulation framework. Thus, in the framework discussed in this section, the GA has been removed, and the dynamic genome pools are replaced by static Blue and Red combat unit genomes.

In the new framework depicted in Figure 22, hand-scripted configuration genomes are constructed in order to represent a Blue unit of networked mobile platforms making up a network-centric combat unit as well as a Red unit of loosely organized, nonnetworked platforms. Combat simulations are run using the `combat_proto` CA engine in combination with the `terrain`, `territory`, and `place` routines to respectively generate a simulation landscape and a territorial importance gradient superimposed over the landscape and to initially position Blue and Red combat agents. Then, the data filter dynamically calculates preselected fitness functions (the entire set of which was described in section 3.1.3) of each opposing combat agent unit within the simulation, which can include hostile IO actions. Finally, the fitness function outputs of the filter are analyzed in a time series format (as opposed to weighting the functions and then combining them into the survival fitness objective function as was done in the previous framework).

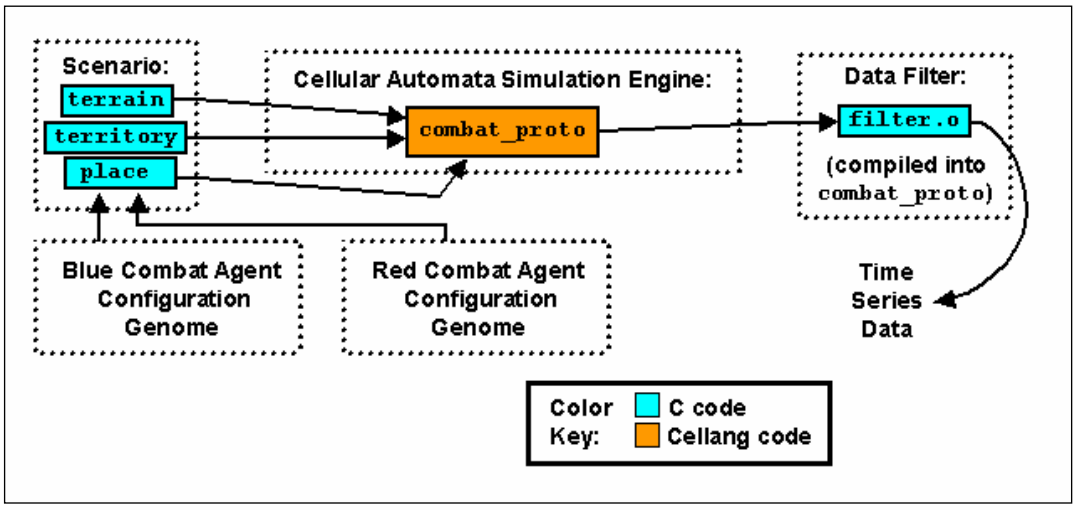


Figure 22. Software framework for running the enhanced Blue/Red quasiscripted combat simulation.

3.2.2 The `combat_proto` CA Simulation Engine

As mentioned in the preceding section, the `combat_proto` CA simulation engine (also written in Cellang [Eckart 1992a, 1992b]) extends the combat agent network-centric capabilities encoded into the `combat_ga` simulation engine by also modeling the deployment and function of notional UGS networks and multihop message relaying between allied agents. As with `combat_ga`, the dynamics of `combat_proto` involve a cyclic sequence of events that is executed during each iterated update of a combat simulation. This sequence is depicted in Figure 23, which illustrates the following cycle of processes (where a process can unfold over one or more sequential simulation time steps):

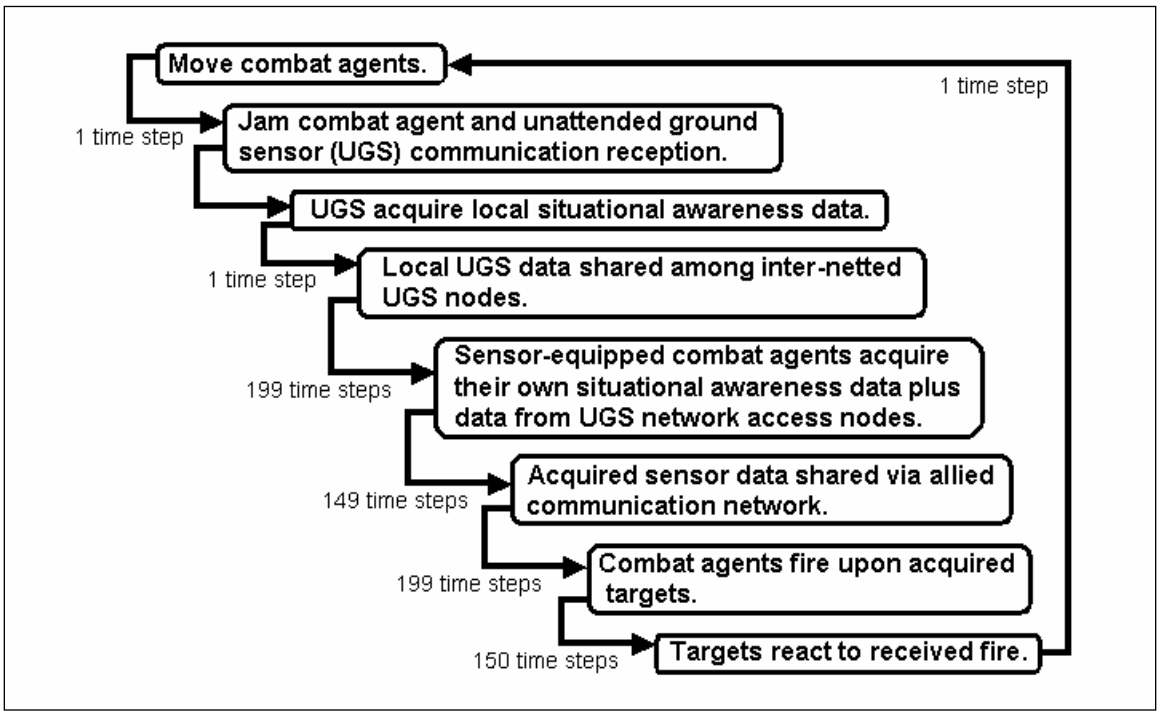


Figure 23. Cyclic dynamics of the `combat_proto` CA simulation engine.

- Combat agents move into an adjacent cell within their local von Neumann neighborhood based on their goal-driven movement decisions (see section 2.3.10.2). This process is executed over one time step.
- Combat agent *and* internetted Blue UGS unit communication reception are jammed based on proximity to previously deployed RF jammer bombs with residual battery power. In this version of the CA engine, (a) combat agents are *not* equipped with on-board jamming capability (i.e., gene G_9 within all agent chromosomes is set equal to 0), and (b) jammer bomb communication disruption is restricted to conventional nine-cell Moore neighborhoods (Figure 24). Next, Blue UGS units sense their local environment (i.e., a single lattice cell). Both of these events occur within a single time step.
- Blue UGS units broadcast their acquired local situational data to other units within the UGS network (based on the UGS dynamics discussed in section 2.3.6.1). This process unfolds over 199 sequential time steps (at the conclusion of which all *connected* portions of the UGS network should share the same SA data).

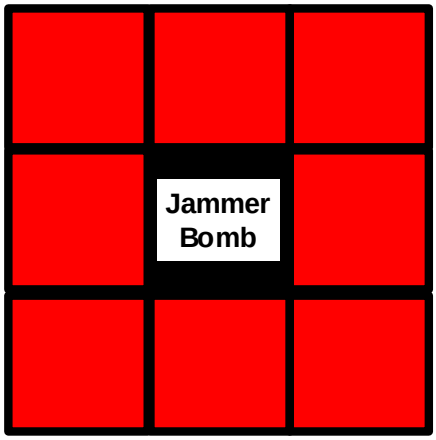


Figure 24. Reduced RF jammer bomb neighborhood consisting of the center bomb cell plus eight neighboring cells.

- Sensor-equipped combat agents acquire SA data transmitted from sensed battlefield entities. In addition, Blue sensor-equipped agents can also acquire SA data from Blue UGS network access nodes.* This data transmission process utilizes artificial "square waves" (which are easy to implement within the regular CA lattice) that travel at a speed of one cell per time step and can be blocked by terrain obstacles such as mountains or buildings. Thus, target sensing by combat agents is assumed to function in a "line-of-sight" mode. This process unfolds over 149 sequential time steps (at the conclusion of which all sensor-equipped agents with uninterrupted line-of-sight pathways to sensed entities should have acquired their data).
- Blue combat agents share acquired sensor data via their allied communication network (while Red agents do not intercommunicate), where an agent will rebroadcast each new data report it receives over the network to ensure that all Blue agents share a coherent battlefield situational awareness. The communication network also transmits data via square waves that travel at a speed of one cell per time step; in this case, however, transmission is *not* limited to line-of-sight interaction (and, thus, terrain obstacles have no effect on message propagation). This process unfolds over 199 sequential time steps (at the conclusion of which all *nonjammed* Blue agents should share the same SA data).
- Combat agents fire upon acquired targets (where multiple agents may engage the same target). Fired rounds travel at a propagation speed of one cell per time step until the target

* The use of specific access nodes within the UGS network was notionally engineered to avoid the cumbersome message saturation that would result if sensor-equipped Blue agents were allowed to acquire stored SA data from all of the nodes within the network.

cell is reached and are also not restricted by line-of-sight between engager and target. This process unfolds over 150 sequential time steps (at the conclusion of which all engaged target agents should have received fire).

- Engaged targets react to received fire (i.e., either die or survive engagement with reduced armor capability). This last process is executed over one time step.

Thus, one entire cycle of simulation processes unfolds over a total of 700 sequential time steps, which are iteratively repeated throughout a combat simulation instance.

3.2.3 "Fombler's Ford"-Inspired Scenario

Figure 25 depicts the virtual battlefield terrain (inspired by the Defense Advanced Research Projects Agency's notional "Fombler's Ford" scenario [Gorman 2000] created to illustrate notional Future Combat Systems operations) within which Blue agents are positioned in the center defending their territory from Red invaders advancing inward from the outer edge of the CA lattice. Associated with this terrain is the territorial "importance" depicted in Figure 26. In this scenario, geographical terrain features such as hills (medium gray), buildings (dark gray regular lattice of cells), a mountain (black), and a river (aqua blue) are added to impact combat agent mobility and UGS placement. As was the case with the importance gradient previously introduced in section 3.1.6.1, the Fombler's Ford gradient shown in Figure 26 is superimposed over the associated terrain and is used to motivate combat agent movement towards a goal (where blue-colored terrain is "neutral," and red-colored terrain represents operationally "critical" territory). Here, combat agent territory unfolds from the "no-man's-land" along the periphery of the CA lattice to a critical Blue-controlled objective point at the center of the 2-D battlespace towards which Red agents advance. Finally, Red-controlled territory is not used in this scenario (and, thus, the Red unit can be thought to represent a collection of loosely organized guerillas or terrorists).

The hand-scripted combat agent chromosomes making up the respective Blue and Red combat unit genomes are shown in Figures 27–32. In this scenario, the Blue unit is composed of the following types of combat agents:

- Three "command and control vehicle" (C2V) agents (Figure 27) equipped with short-range sensor, short-range firepower, and long-range interagent communication capabilities, as well as a nonzero value of the human crew gene G_{16}. Since the only autonomous C2 capability built into this type of agent is to allow other Blue agents without human crew to engage acquired targets, the C2V agents are functionally redundant (i.e., they do not control specific teams within the Blue unit). Thus, "unmanned" Blue firepower-equipped agents can engage targets provided (1) at least one C2V agent survives and (2) the engaging agent's communication reception remains unjammed. Finally, these agents,

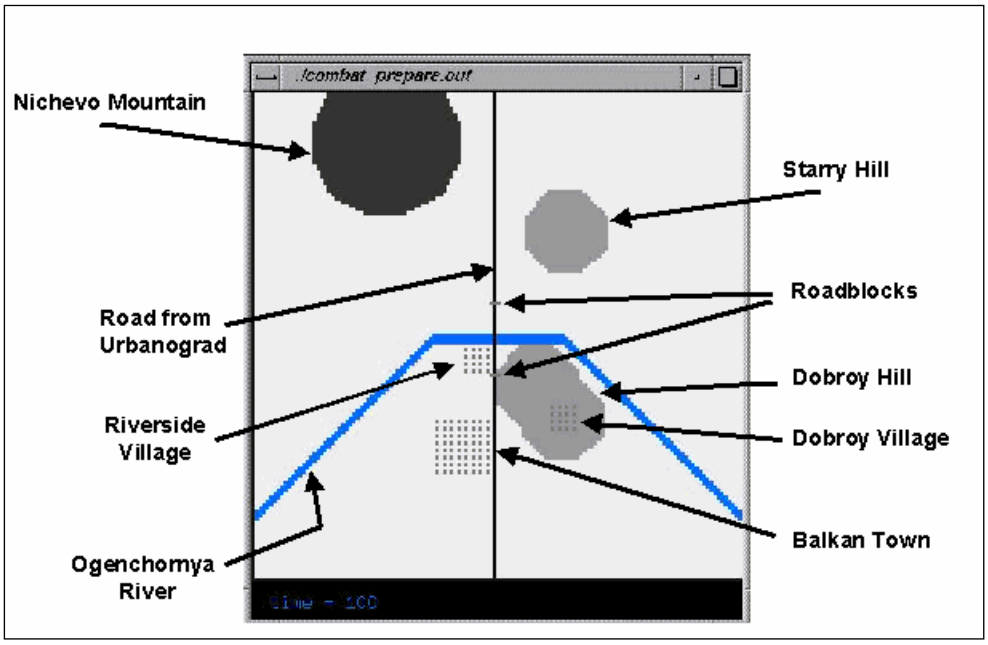


Figure 25. Notional Fombler's Ford terrain imposed onto the CA lattice.

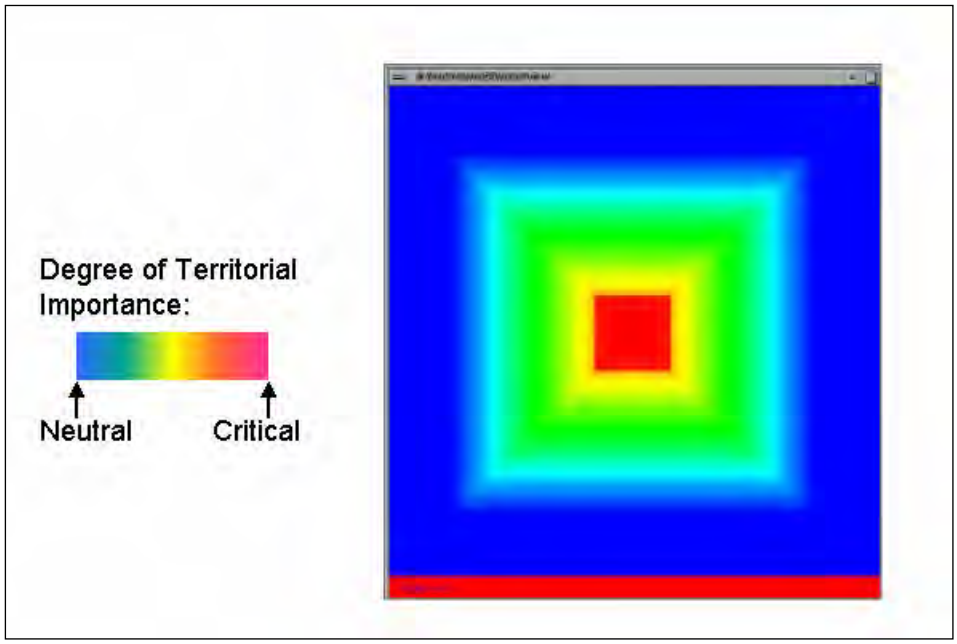


Figure 26. Territorial importance gradient associated with the Fombler's Ford-inspired combat scenario.

although designated as "ground vehicles," are *not* given mobility capability in this scenario (since they were specifically designed to maintain a static defensive posture throughout the duration of a combat simulation instance).

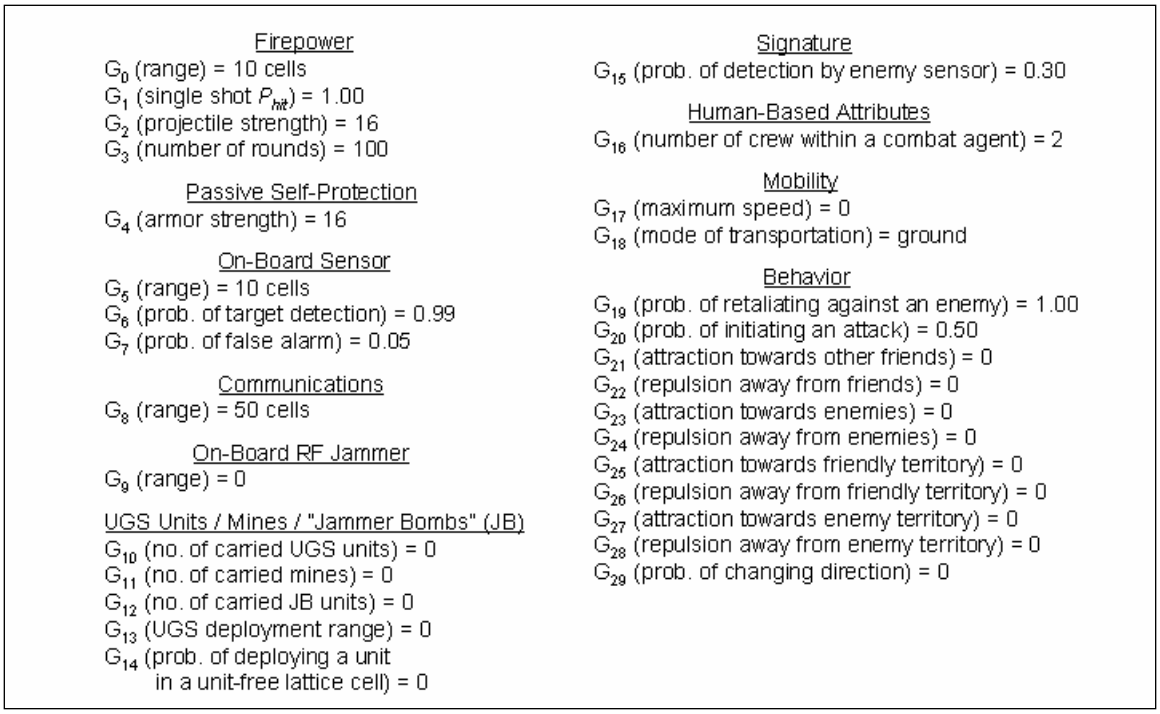


Figure 27. Blue C2V agent chromosome.

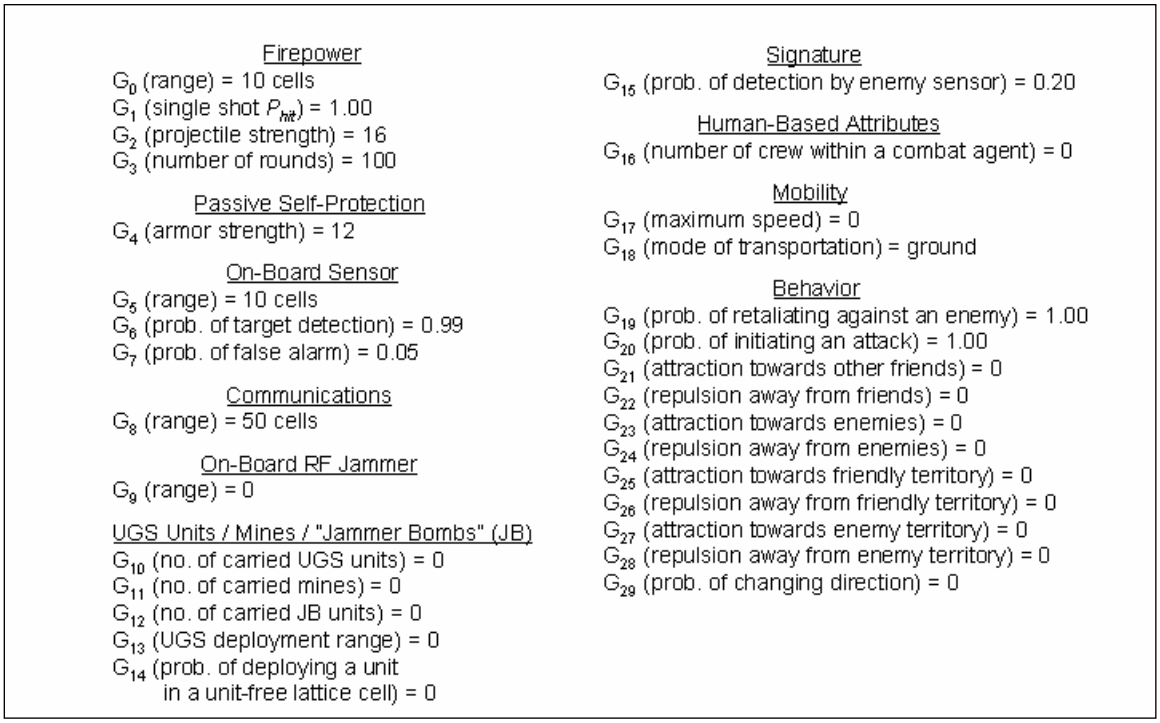


Figure 28. Blue short-range direct-fire (DF) agent chromosome.

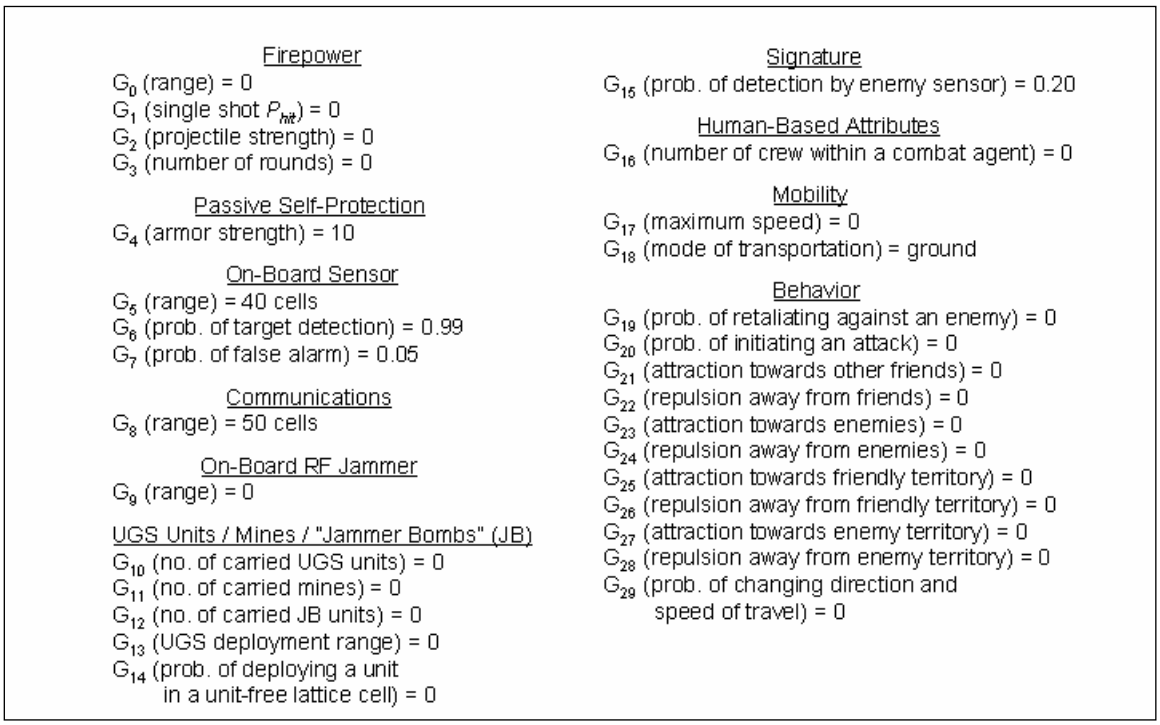


Figure 29. Blue long-range sensor agent chromosome.

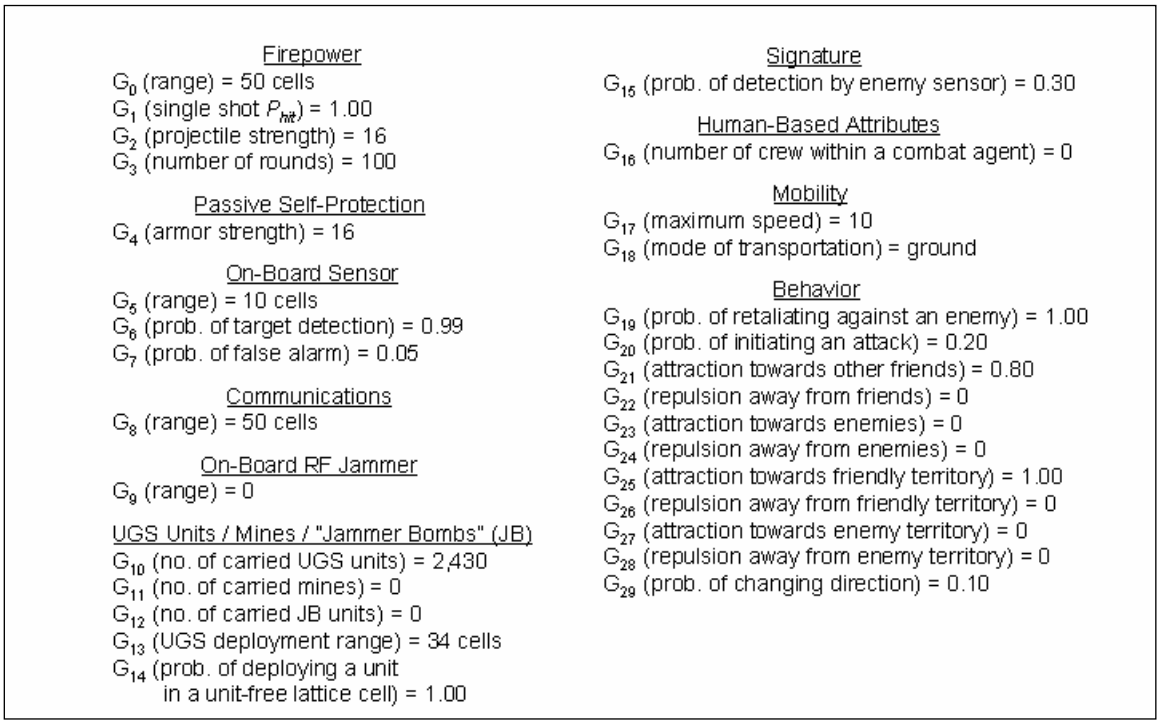


Figure 30. Blue “rockets-in-a-box” agent chromosome.

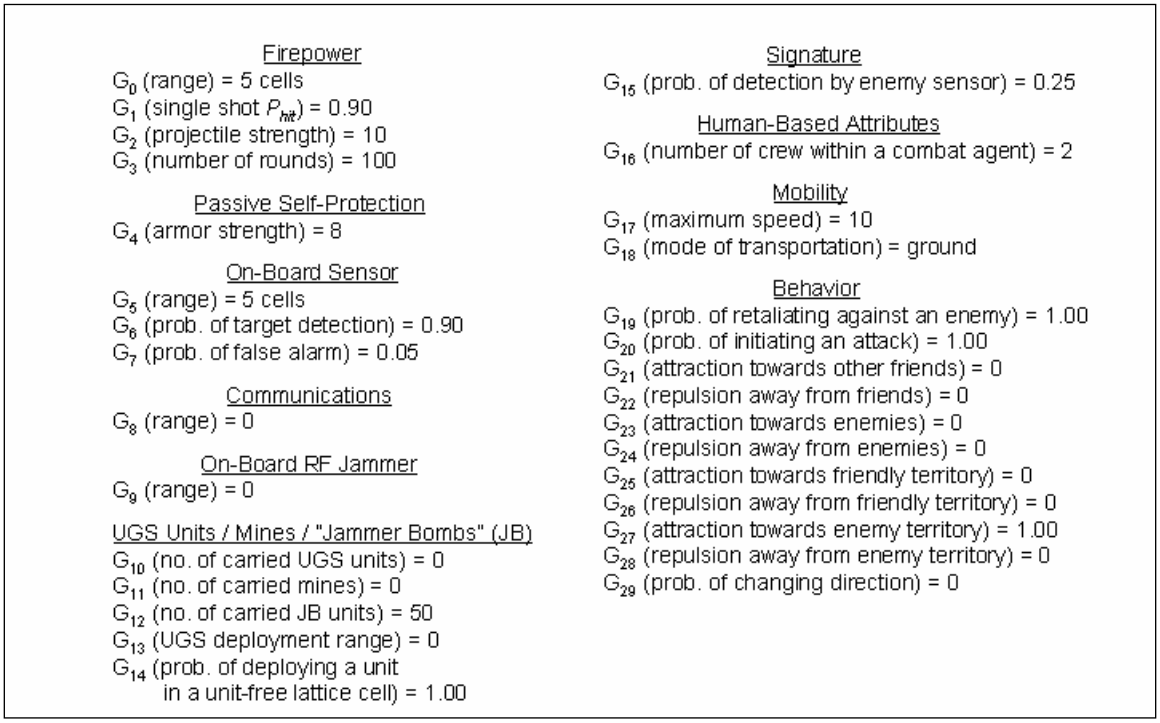


Figure 31. Red "pickup truck" agent chromosome.

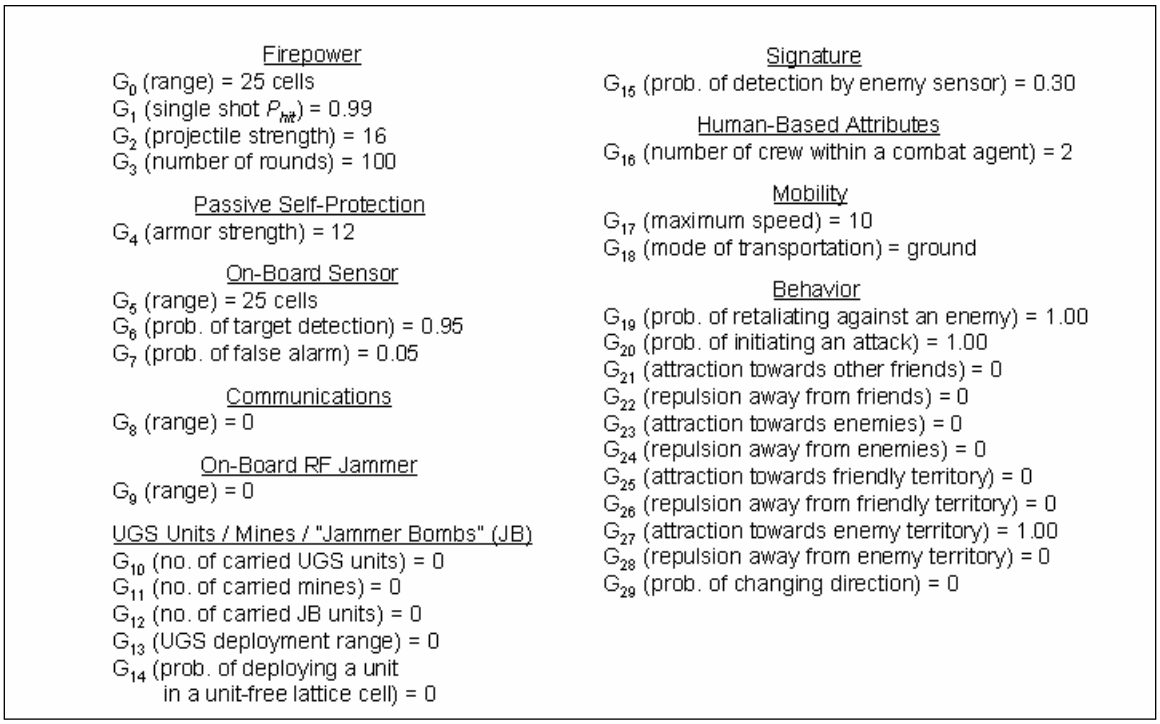


Figure 32. Red medium-range DF agent chromosome.

- Three C2V agents (Figure 27) equipped with short-range sensor, short-range firepower, and long-range interagent communication capabilities, as well as a nonzero value of the human crew gene G_{16}. Since the only autonomous C2 capability built into this type of agent is to allow other Blue agents without human crew to engage acquired targets, the C2V agents are functionally redundant (i.e., they do not control specific teams within the Blue unit). Thus, "unmanned" Blue firepower-equipped agents can engage targets provided (1) at least one C2V agent survives and (2) the engaging agent's communication reception remains unjammed. Finally, these agents, although designated as "ground vehicles," are *not* given mobility capability in this scenario (since they were specifically designed to maintain a static defensive posture throughout the duration of a combat simulation instance).
- Two DF agents (Figure 28) equipped with the same capabilities assigned to the C2V agent *except* for the lack of a human crew (i.e., this DF agent type represents an unmanned ground vehicle). This type of agent also maintains a static defensive posture throughout a simulation run and thus serves as a "sacrificial diversion" that draws enemy fire away from more important targets.
- Three unmanned sensor agents (Figure 29) equipped only with long-range sensor and communication capabilities. These agents also maintain static defensive postures throughout a simulation run.
- Four rockets-in-a-box agents (Figure 30) equipped with short-range sensor, long-range firepower, long-range communication, *and* mobility capabilities, as well as a plentiful supply of 270 UGS missiles (with nine deployable UGS units packed into a missile). These agents are provided with mobility to facilitate their deployment of the Blue UGS network (see the next paragraph), where agent movement is motivated by attraction to a combination of Blue agent CM and Blue territory (to move into their defensive postures after UGS network deployment). Finally, gene G_{20} (i.e., probability of initiating an attack) is set equal to 0.20 for all of these agents to imply an inherent difficulty in distinguishing viable Red targets apart from indigenous "neutral" agents (which are assumed to be present but are not explicitly modeled in the scenario).

The total Blue force thus consists of 12 different combat agents drawn from 4 different agent classes or "species."

On the other hand, the Red unit is composed of the following types of combat agents:

- Twenty pickup truck agents (Figure 31) equipped with very short-range sensor (i.e., human vision provided by the "driver" and "passenger" riding in the truck bed), very short-range firepower (i.e., a machine gun belonging to the truck's "passenger"), and mobility capabilities, as well as a nonzero value of the human crew gene G_{16} and a supply of RF jammer bombs (that will be deployed from the truck bed by the "passenger"). These agents

do not intercommunicate or communicate with other Red agents and thus function in an operationally autonomous fashion. Finally, agent movement is motivated only by an attraction to Blue territory.

- Ten DF agents (Figure 32) equipped with medium-range sensor, medium-range firepower, and mobility capabilities. These agents also do not intercommunicate or communicate with Red pickup truck agents and thus are operationally autonomous. Unlike the Blue rockets-in-a-box agents, these DF agents have no difficulty in positively identifying enemy agents once acquired. Finally, agent movement is motivated only by an attraction to Blue territory.

The total Red force thus consists of 30 different combat agents drawn from two different agent species.

As was previously mentioned, a C2 process was directly built into the `combat_proto` CA simulation engine that guides combat agent activities via a semiscripted time-dependent schedule called a synchronization matrix (Figure 33). In this C2 process, each specific type or class of Blue and Red combat agent is identified, along with a schedule of combat "orders" assigned to that agent type as a function of mission time (represented by the time axis below the matrix). The C2 activity commences some 830 time steps prior to mission commencement. Here, Blue rockets-in-a-box agents equipped with "UGS missiles" fire UGS units into place within the neutral terrain surrounding Blue territory, while other Blue agents maintain a stationary defensive posture. This UGS network deployment process is depicted in Figure 34 as a sequence of simulation window snapshots. Each of the four Blue rockets-in-a-box agents is initially positioned in the center of a battlefield quadrant, from which vantage point they launch the UGS missiles. Upon landing at a target cell, a UGS missile acts to populate a 3 × 3 cell sublattice with internetted UGS units following an average 75% packing density (i.e., the probability that a cell within the nine-cell UGS sublattice receives a UGS unit is equal to 0.75).* Once the UGS network is in place, the rockets-in-a-box agents move to defensive postures and remain there. Then, at time $t = 0$, a "first wave" of the Red seemingly-neutral pickup truck agents (many of which are acquired and fired upon/destroyed by the network-centric Blue unit) drop jammer bombs as they penetrate into the peripheral Blue UGS network. Finally, at time step $t = 4200$ (i.e., the commencement of the seventh simulation time cycle), a "second wave" of the Red agents equipped with medium-range firepower capability advances toward Blue in the undetected "shadow zones" within the UGS network created by the dispersed jammer bombs. The simulation is then allowed to run until a steady-state end condition is achieved (i.e., either Blue successfully kills all Red invaders or all surviving Red agents reach the center of the Fombler's Ford terrain).

* The UGS packing density is a variable control parameter that will be utilized in a sensitivity analysis presented in a subsequent section of the report.

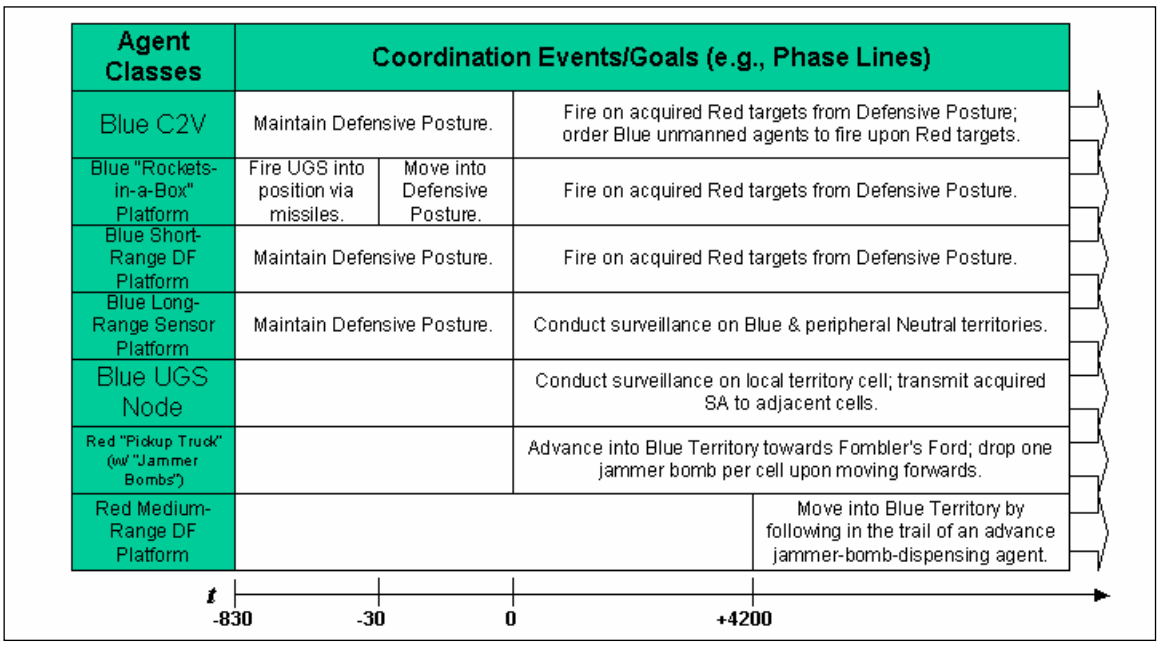


Figure 33. Synchronization matrix for the Fombler's Ford-inspired combat scenario.

Starting at $t = 0$ and then moving forward in time, a combat simulation is instantiated by running the `combat_proto` CA engine in combination with the Fombler's Ford terrain, associated territorial importance gradient, and the synchronization matrix. Figure 35 depicts the resultant events or phases that unfold throughout one simulation update cycle of a CA simulation run. In Figure 35a, Red combat agents enter the peripheral neutral territory surrounding Blue territory and are sensed by the notional UGS network (blue cells). Once a UGS unit senses a combat agent that has entered its cell, it transmits agent identity (Red) north/south/east/west neighboring UGS units (represented by the flowing green "message fronts"); this allows for situational awareness reports to propagate throughout connected portions of the extensive UGS "web." In Figure 35b, sensor messages are sent out by UGS "access nodes" and Blue agents; these messages are then "sensed" and received by sensor-equipped combat agents. In the next snapshot (Figure 35c), sensor data are shared among Blue agents via their communication network. Sensor and communication messages travel as artificial "square waves" to ensure symmetric signal propagation throughout the CA lattice. Finally, as shown in Figure 35d, a Blue agent fires upon an acquired Red target (where the black line illustrates payload trajectory).

The propagation dynamics of locally sensed SA data throughout the deployed UGS network is illustrated in greater detail as a sequence of simulation window snapshots in Figure 36. In this simulation instance, Red pickup truck agents enter the outer edges of the UGS network from the north, south, east, and west. Then, as time progresses, local UGS data are shared across connected network nodes (where the network cell color indicates the total number of SA data

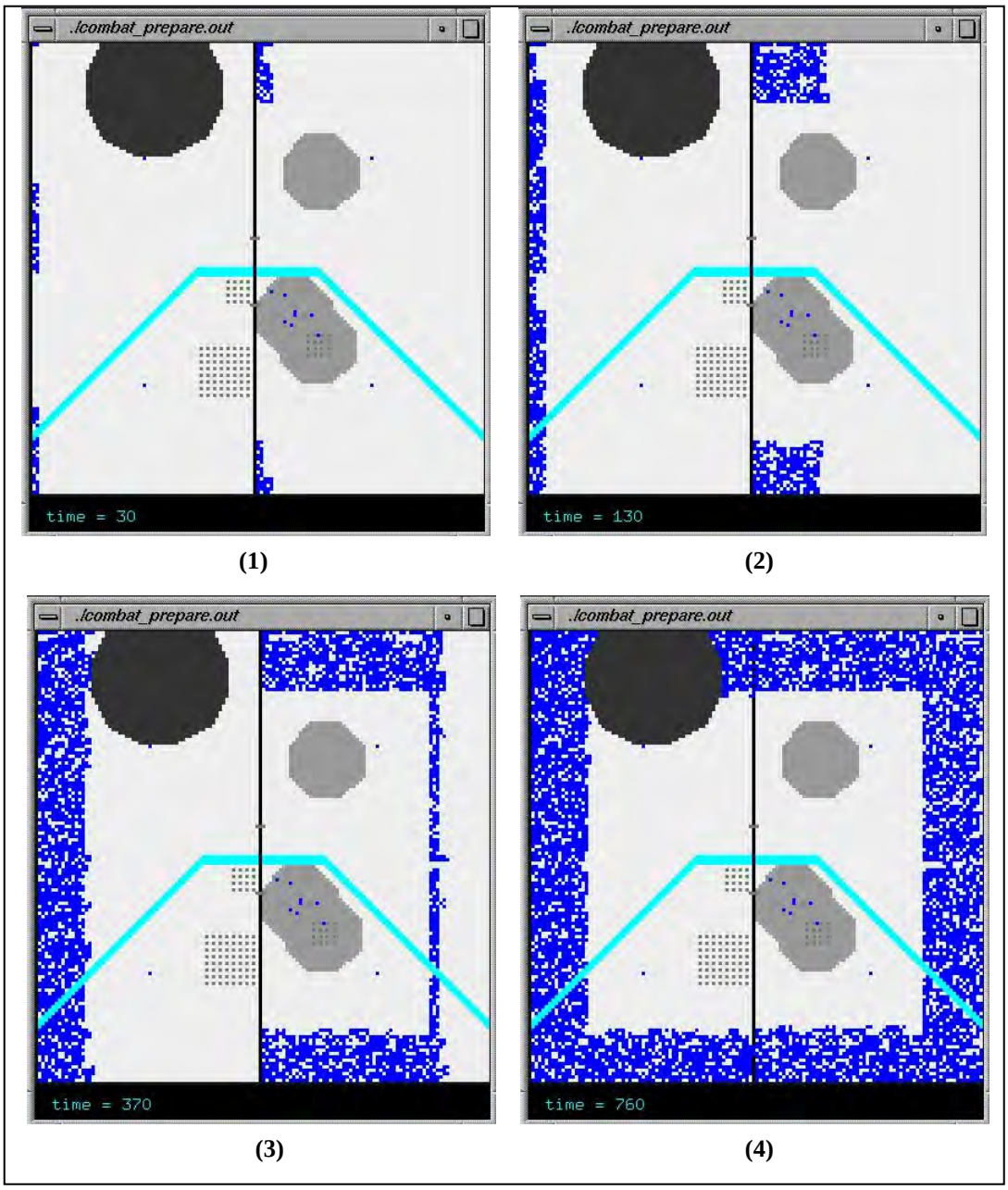


Figure 34. Four successive snapshots illustrating the deployment of a notional UGS network across neutral territory within the Fombler's Ford landscape.

messages resident within a node at that point in time). A final equilibrium condition is achieved within the network when all interconnected UGS nodes possess the same set of SA data messages (time snapshot no. 4 in Figure 36). In the current UGS network configuration, there are three distinct disconnected subnetworks that emerge as a function of terrain features (i.e., Nichevo Mountain and the Ogenchornya River) that naturally subdivide the overall network.

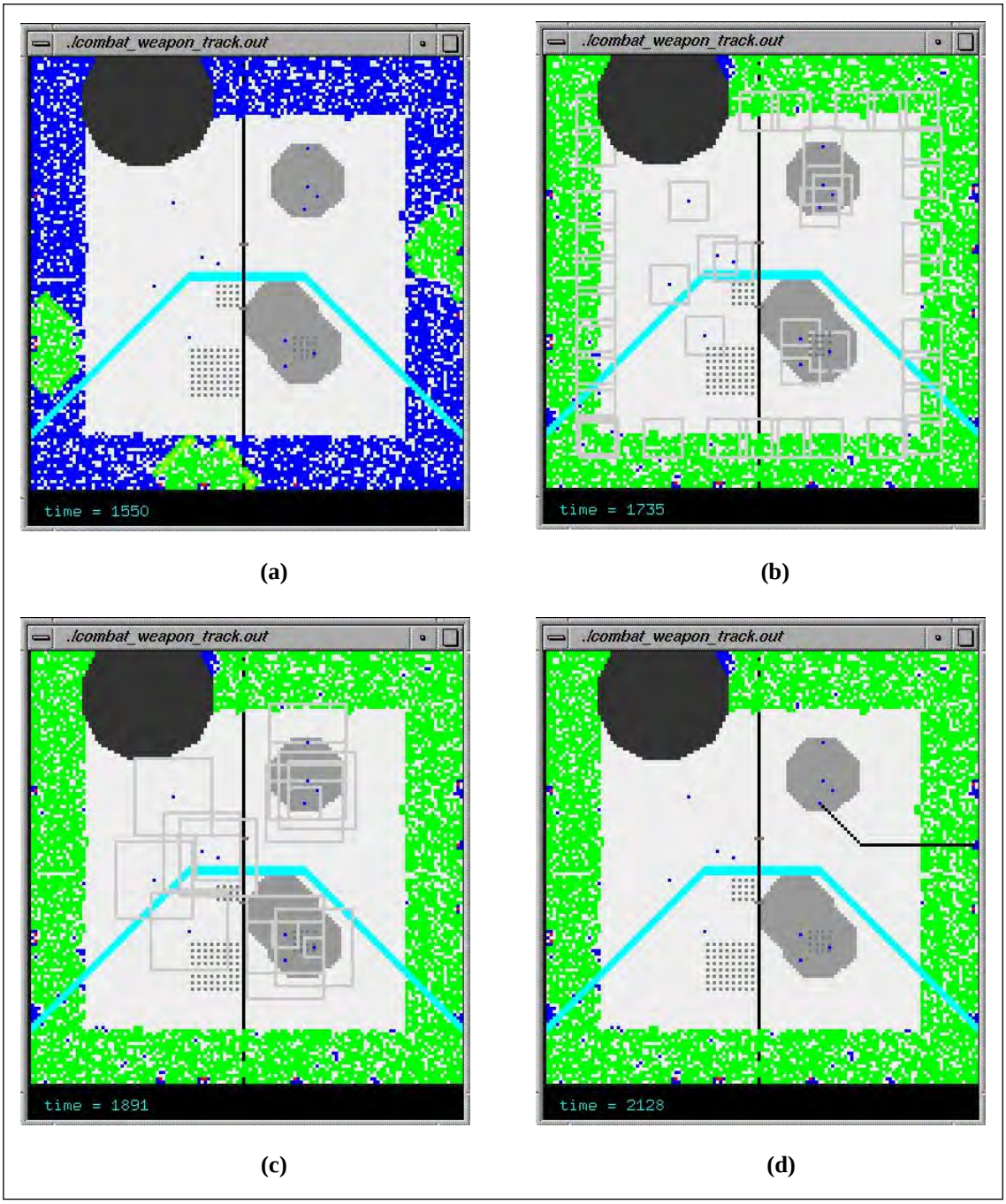


Figure 35. Four phases within a `combat_proto` simulation cycle using the Fombler's Ford-inspired scenario: (a) Red agents detected within the peripheral UGS network, (b) sensor data propagates from sensed entities to sensor-equipped Blue agents, (c) acquired sensor data are shared over the Blue communication network, and (d) Blue fires on Red target.

The previously discussed UGS network packing density, which is directly correlated with deployed UGS web interconnectivity, is related to site percolation. A concept borrowed from

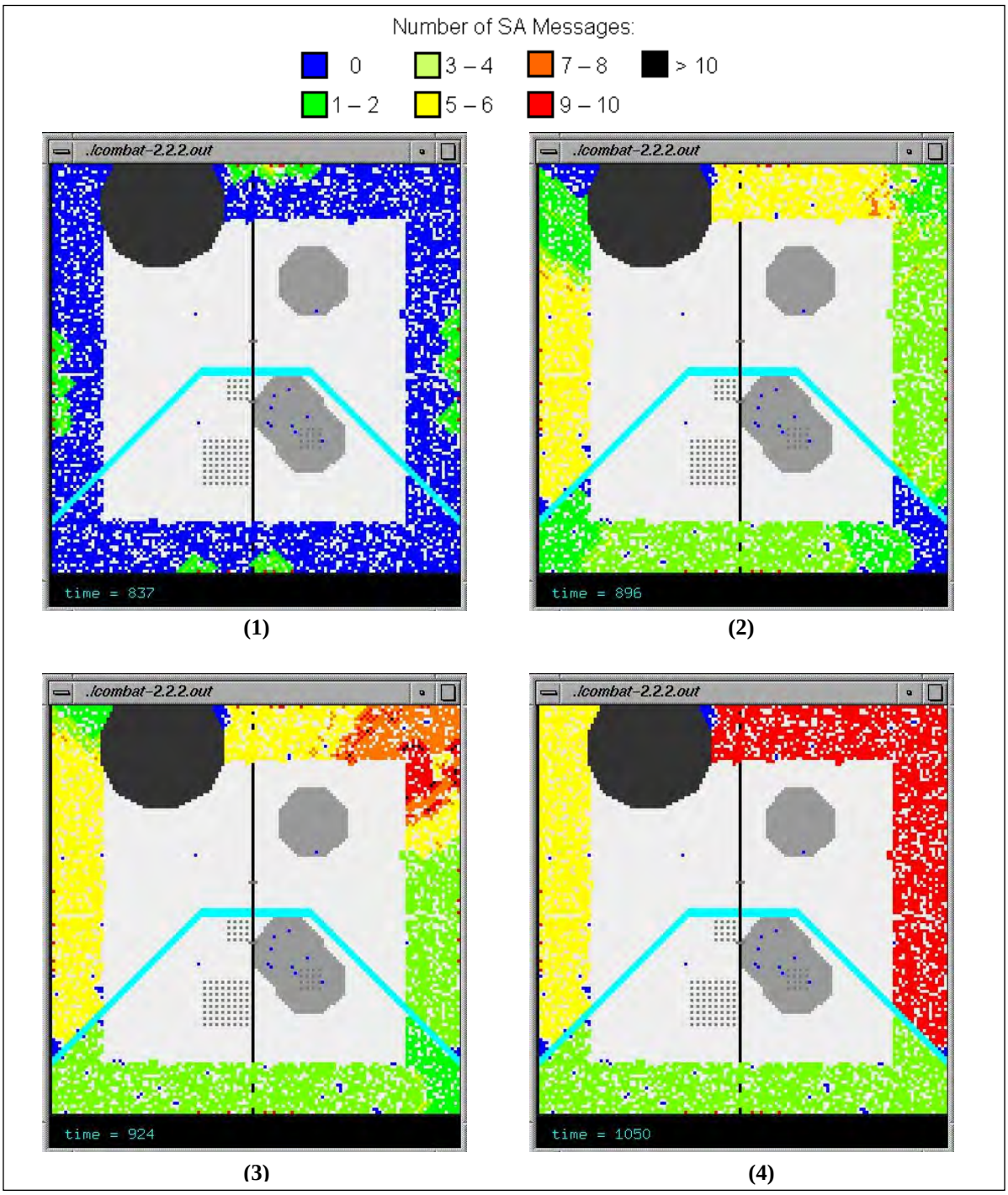


Figure 36. Four successive snapshots illustrating the propagation dynamics of SA data throughout the UGS network.

statistical mechanics, site percolation refers to the uninterrupted flow of a resource (in this case, information) through a contiguous medium comprised of nodes within a 2-D regular lattice and is known to emerge in certain types of CA with heterogeneous cell transition rules (Wolfram 1985). Associated with this process is a critical packing density threshold, $p_c \cong 0.593$, at which

point the information flow between nodes changes from locally to globally continuous (Ziff 1992). Thus, the UGS network packing density, which is meant to represent factors such as local environment metrology or terrain that might tend to disrupt orderly UGS unit deployment, should always be maintained at a level well above p_c.

Finally, in addition to setting up the UGS network across the neutral territory surrounding the central Blue territory prior to mission commencement, the Blue unit is also assumed to have configured a field of land mines along the outer edges of its territory prior to commencement of a simulation run. This preconfigured minefield is depicted in Figure 37. In the "unlikely" event of Red agents reaching the periphery of Blue territory, the minefield is intended to prevent further penetration of invader agents via immobilization (see section 2.3.6.2). Since the probability of a Red agent losing its mobility capability upon entering a mined cell is equal to 0.5 and the minefield is three cells in width, the probability of the agent getting through the minefield with mobility intact is $0.5^3 = 0.125$.

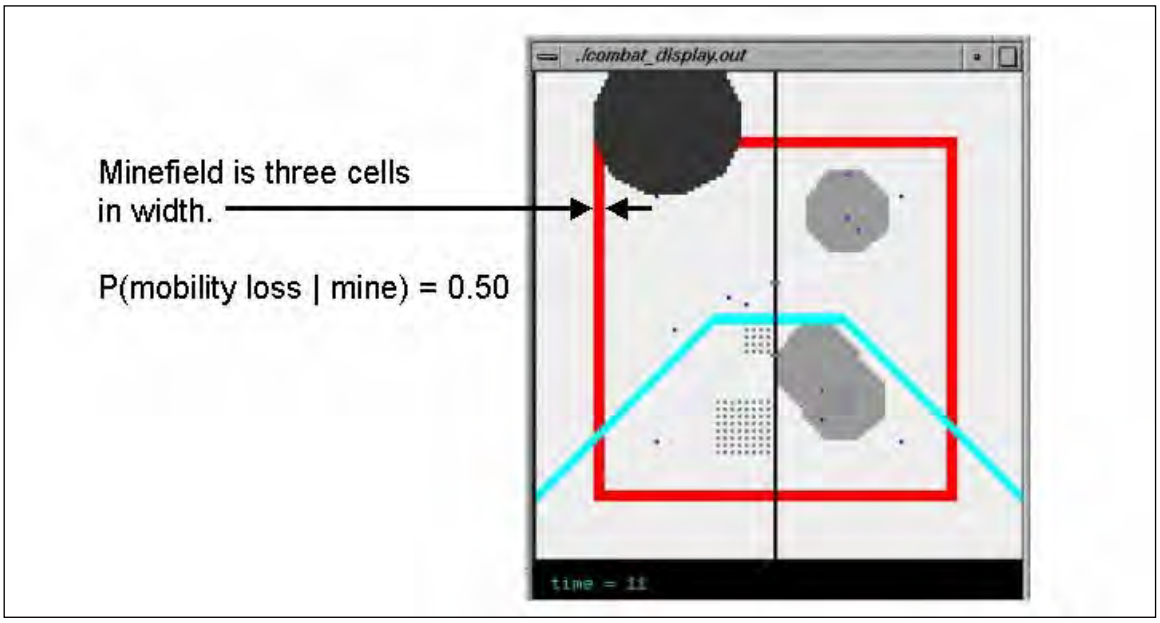


Figure 37. Preconfigured land mine field surrounding Blue territory.

The principal objective of the Red unit in this combat scenario is to allow their medium-range DF agents to infiltrate through the UGS network and reach the periphery of Blue territory without being detected (from which vantage point the Red DF agents can engage Blue targets). Thus, the intent of Red is to create unsensed "rips" within Blue's UGS network (through jammer bomb deployment) through which the Red medium-range DF agents may move undetected. This "sensor web rip" operation by Red is illustrated in Figure 38, which depicts the channels within the UGS network wherein communication reception has been jammed during the 12th simulation time cycle. Here, the color of the UGS network nodes again indicates the number of

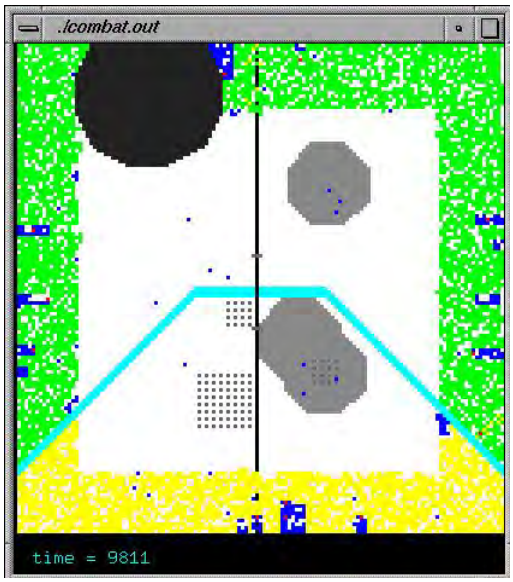


Figure 38. The status of Red's sensor web rip operation by the end of the 12th time cycle within a `combat_proto` simulation instance.

SA data messages resident in a node (see Figure 36 for an explanation of the color code). Given that jammer bomb battery lifetimes have been preset to a duration of 10 successive time cycles subsequent to bomb deployment, the rips in the UGS network are sufficiently perpetuated to allow the Red medium-range DF agents ample opportunity to reach the edge of Blue territory.

3.2.4 Simulation Results

Combat agent simulations were run using the software framework depicted in Figure 22 in combination with the Fombler's Ford scenario described in section 3.2.3 in order to explore the sensitivity of the following model parameters to IO stress:

- UGS density - the average packing density of the peripheral Blue UGS network.
- Blue fire point - the outward distance from the perimeter of Blue territory at which point Blue combat agents are allowed to fire on acquired Red targets.
- Blue shots/cycle - the number of shots a firepower-equipped Blue combat agent can fire during one simulation time cycle.

This sensitivity analysis was predicated on the time series response generated by 4 of the 10 fitness functions (introduced in section 3.1.3) as applied to the following specific combat units:

- Blue and Red combat unit survival ($F_3[t_n]$; equation 13),
- Blue's ability to sense the battlefield ($F_5[t_n]$; equation 15),

- Blue's ability to prevent Red penetration into Blue territory with dynamic renormalization ($F_7[t_n]$; equation 17), and
- collective Blue situational awareness of Red ($F_{10}[t_n]$; equation 20).

In this set of simulations, these fitness functions serve as dynamic measures of collective combat unit performance.

Time series results from the first of six different Fombler's Ford scenario variants are depicted in Figures 39 and 40. The purpose of this initial scenario variant is to illustrate the operational impact within an IO-stress-free environment (and, thus, Red "pickup truck" agents do *not* deploy jammer bombs in this variant) of Blue deploying a UGS network with a packing density less than the critical site percolation density p_c (see section 3.2.3). In this scenario, the model parameters are set to (a) UGS density = 0.55, (b) Blue fire point = 10 cells, and (c) Blue shots/cycle = 1. Multiple simulation instances of the scenario are run using the set of 20 sequential random seeds {0, 1, 2, …, 19},* and then time series results are averaged and plotted. The Blue unit's average fractional survival (Figure 39a) is significantly reduced to a post-combat level of 0.63, while the Red unit's average survival rapidly drops to a near-zero level following the commencement of combat (and then soon achieves total attrition by time cycle 41). Blue's average fractional sensor coverage of the virtual 2-D battlespace (Figure 39b) is reduced from ~0.86 to a bit higher than 0.67 through the loss of sensor-equipped Blue agents. The Blue unit's average "defender fitness" capability to prevent Red penetration into Blue territory (Figure 40c) also seriously underperforms as a result of the subcritical UGS network packing density, where Red medium-range DF agents are able to penetrate the periphery of Blue territory (starting at time cycle 20) and significantly engage Blue targets until the Red agents are killed off (by time cycle 41). Finally, the Blue unit's average situational awareness of Red (Figure 40d) remains below a level of 0.70 out until about time cycle 30 (again attributable to the subcritical UGS density), by which time enough Red agents reach locations at the edge of Blue territory where engagement of Blue targets is possible.

In the second Fombler's Ford scenario variant, the UGS network packing density is raised to a level well above p_c, while other parameters remain as before. Simulation results from this scenario (again averaged over 20 simulation instances using the standard set of random seeds) are depicted in Figures 41 and 42. Thus, in these runs, the model parameters are now set to (a) UGS density = 0.75, (b) Blue fire point = 10 cells, and (c) Blue shots/cycle = 1. Also, in order to quantitatively illustrate the dynamic impact of IO stress on both Blue and Red combat unit operational performance, two sets of simulations are run using these model parameters. In the first set, Red pickup truck agents operate without jammer bombs. In the second set, these Red agents are each equipped with a supply of jammer bombs, as delineated in Figure 31.

* Since this sequence of 20 random seeds will also be used in subsequent scenario variant simulation runs for consistency purposes, it will henceforth be referred to as the "standard set of random seeds."

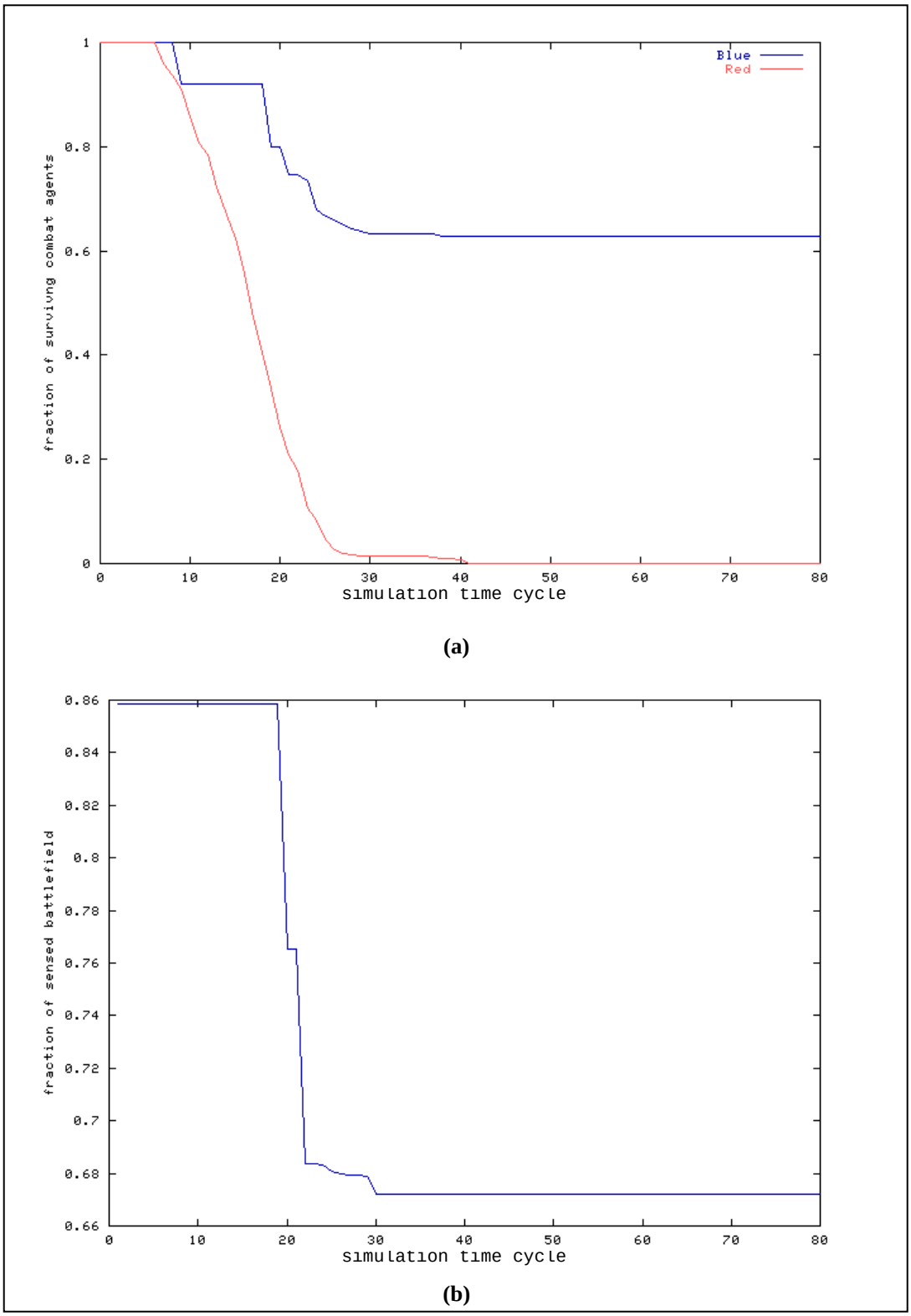


Figure 39. Fombler's Ford simulation results, with UGS density = 0.55, Blue fire point = 10 cells, and 1 Blue shot/cycle: (a) combat unit survival and (b) Blue sensor coverage.

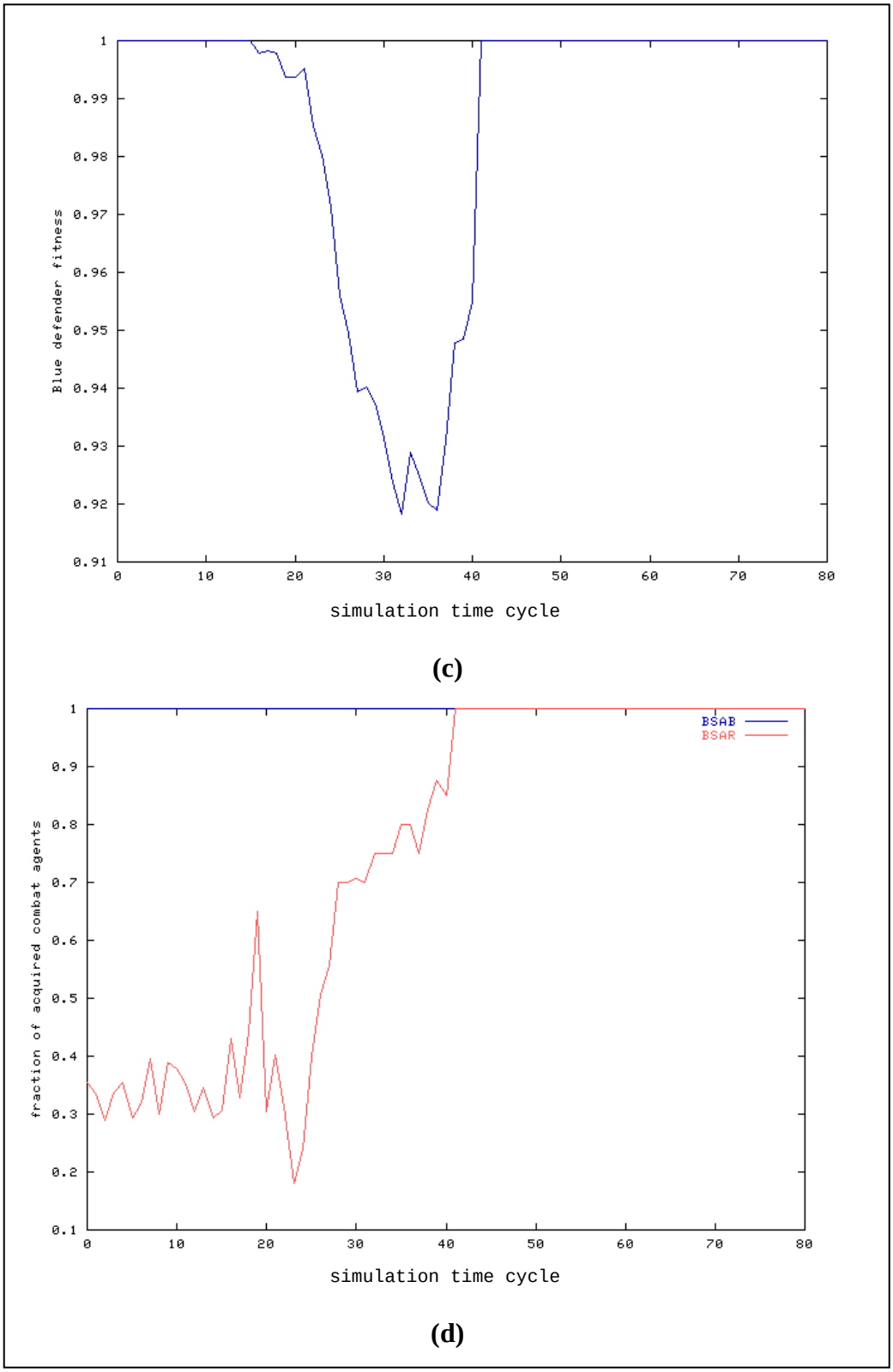


Figure 40. Fombler's Ford simulation results, with UGS density = 0.55, Blue fire point = 10 cells, and 1 Blue shot/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

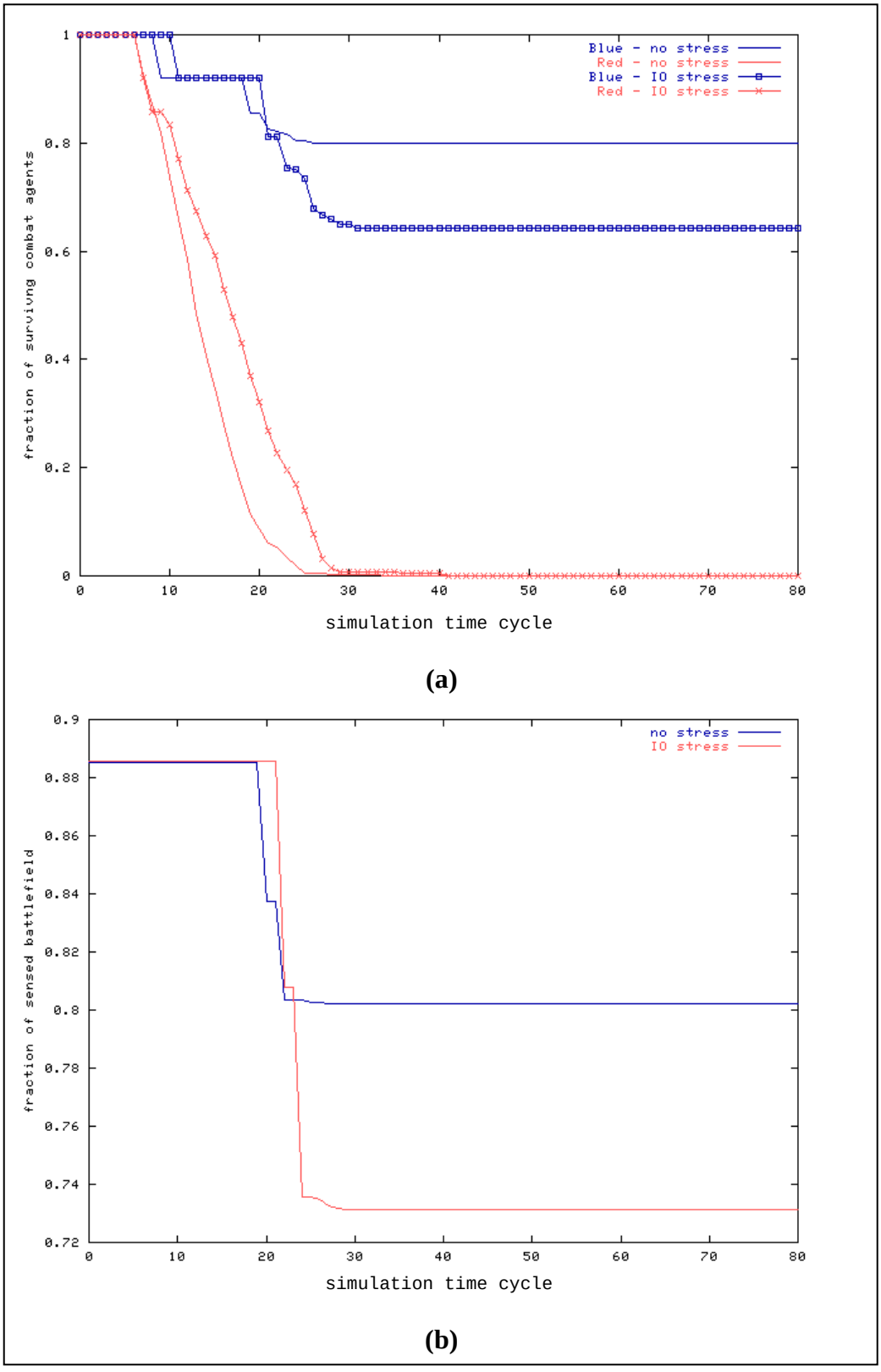


Figure 41. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 10 cells, and 1 Blue shot/cycle: (a) combat unit survival and (b) Blue sensor coverage.

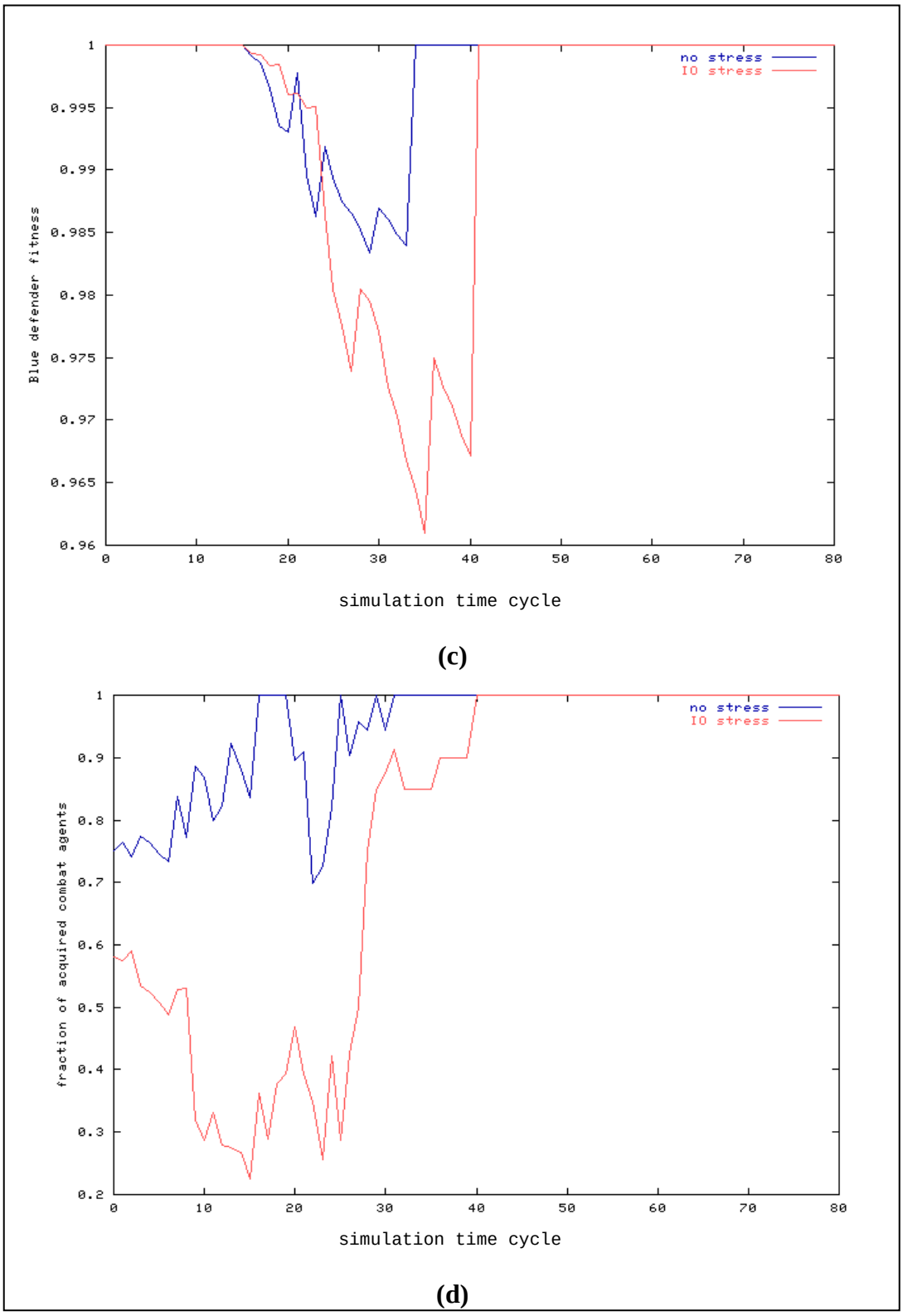


Figure 42. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 10 cells, and 1 Blue shot/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

As one would expect, the operational impact of simply increasing Blue's UGS packing density from 0.55 to 0.75 is considerable. Blue's average post-combat survival level (Figure 40a) *within an IO-benign environment* increases to 0.80, while Red's attrition rate is slightly increased. The post-combat fractional Blue sensor coverage (Figure 41b) also increases to a bit over 0.80. Blue's average defender fitness level (Figure 42c) only degrades to a level of ~0.985 (indicating a reduction in Red's ability to penetrate Blue territory). Finally, Blue's situational awareness of Red (Figure 42d) is sustained at levels greater than 0.70.

However, the addition of IO stress (by equipping the Red pickup truck agents with jammer bombs) is seen to significantly degrade the performance of all of the time series metrics depicted in Figures 41 and 42. Blue's post-combat survival level is reduced by almost 25% under stress, while Red's survival increases shortly the after the second invasion wave commencement at the seventh time cycle. Blue's sensor coverage is reduced by ~10% under stress. Blue's defender fitness is slightly degraded by the IO stress, with Red agents able to remain within the outer regions of Blue territory for an additional 7–8 simulation time cycles over the unstressed occupation time duration. Lastly, Blue's situational awareness of Red is significantly degraded by the rips induced within UGS web continuity via the Red jammer bomb deployment, essentially returning to levels observed in the previous scenario variant.

In the third Fombler's Ford scenario variant, Blue's fire point is increased so that Red targets are now engaged as soon as they are introduced at the edge of the CA lattice (with other parameters remaining as before). Simulation results from this scenario (again averaged over the standard set of random seeds) are depicted in Figures 43 and 44. In these runs, the model parameters are next set to (a) UGS density = 0.75, (b) Blue fire point = 17 cells, and (c) Blue shots/cycle = 1; again, two sets of simulations are run (with and without jammer bombs). Here, Blue's average *unstressed* post-combat survival level (Figure 43a) increases to 0.92, with Red's attrition rate even slightly greater than in the previous scenario; again, the addition of IO stress degrades the former performance and improved the latter. The post-combat fractional Blue sensor coverage (Figure 43b) increases to a level of ~0.88 without stress and then drops to 0.837 under stress. Next, Blue's average defender fitness (Figure 44c) virtually remains at ~1.00 without IO stress and is then degraded down to similar levels as seen in the previous scenario variant upon the introduction of stress. Finally, unstressed Blue situational awareness of Red (Figure 44d) is sustained at levels now greater than 0.80; again, these levels drop significantly when Red agents are allowed to use their jammer bombs.

In the fourth Fombler's Ford scenario variant, Blue's fire point is reduced to its original value while that unit's firepower engagement rate is increased. Simulation results from this scenario (averaged over the standard set of random seeds) are depicted in Figures 45 and 46. In these runs, the model parameters are next set to (a) UGS density = 0.75, (b) Blue fire point = 10 cells, and (c) Blue shots/cycle = 3; again, sets of simulations with and without IO stress are run. In this case, Blue's average performance within both unstressed and IO-stressed environments regarding post-combat survival level (Figure 46a), fractional sensor coverage (Figure 45b),

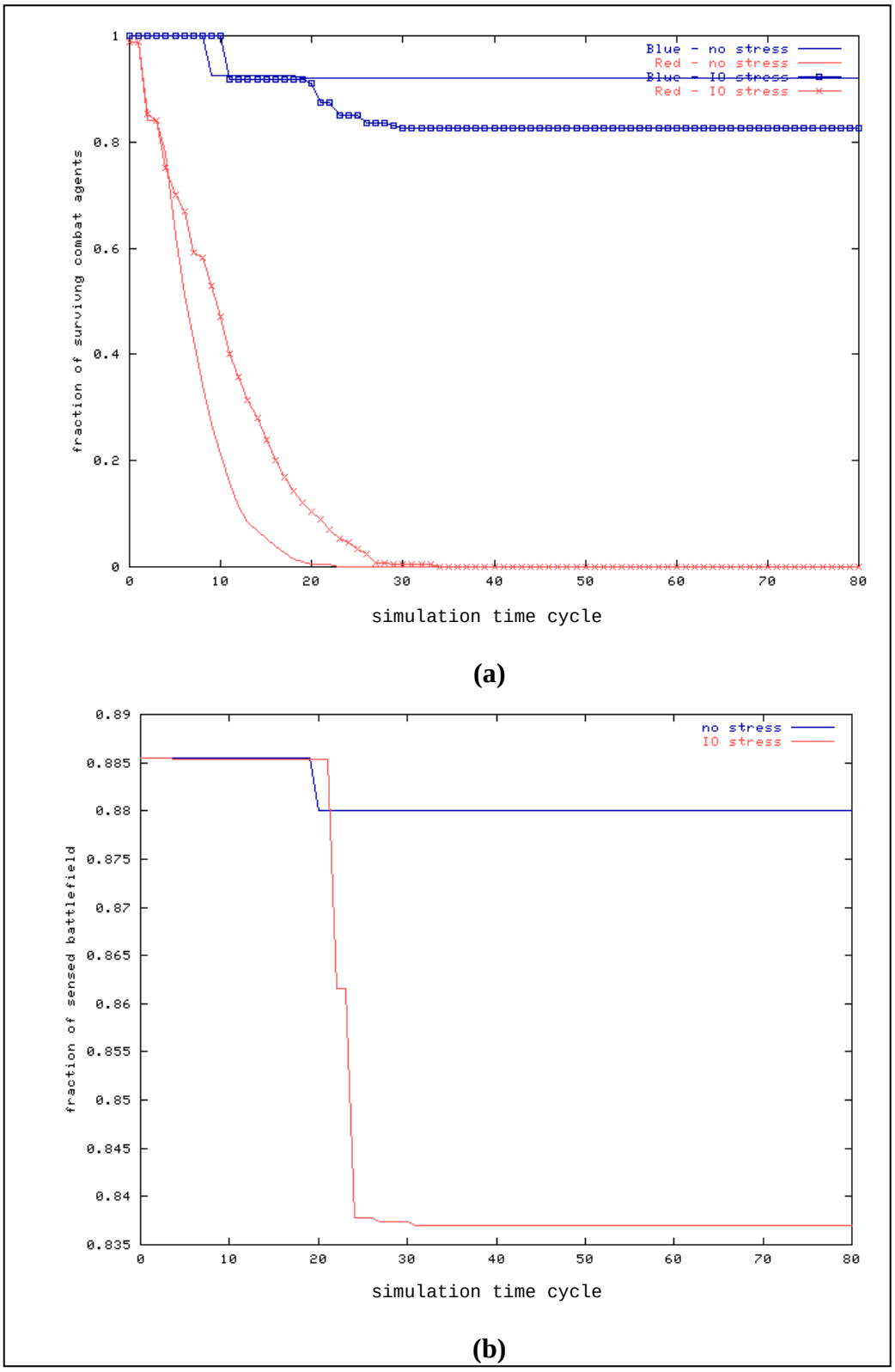


Figure 43. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 17 cells, and 1 Blue shot/cycle: (a) combat unit survival and (b) Blue sensor coverage.

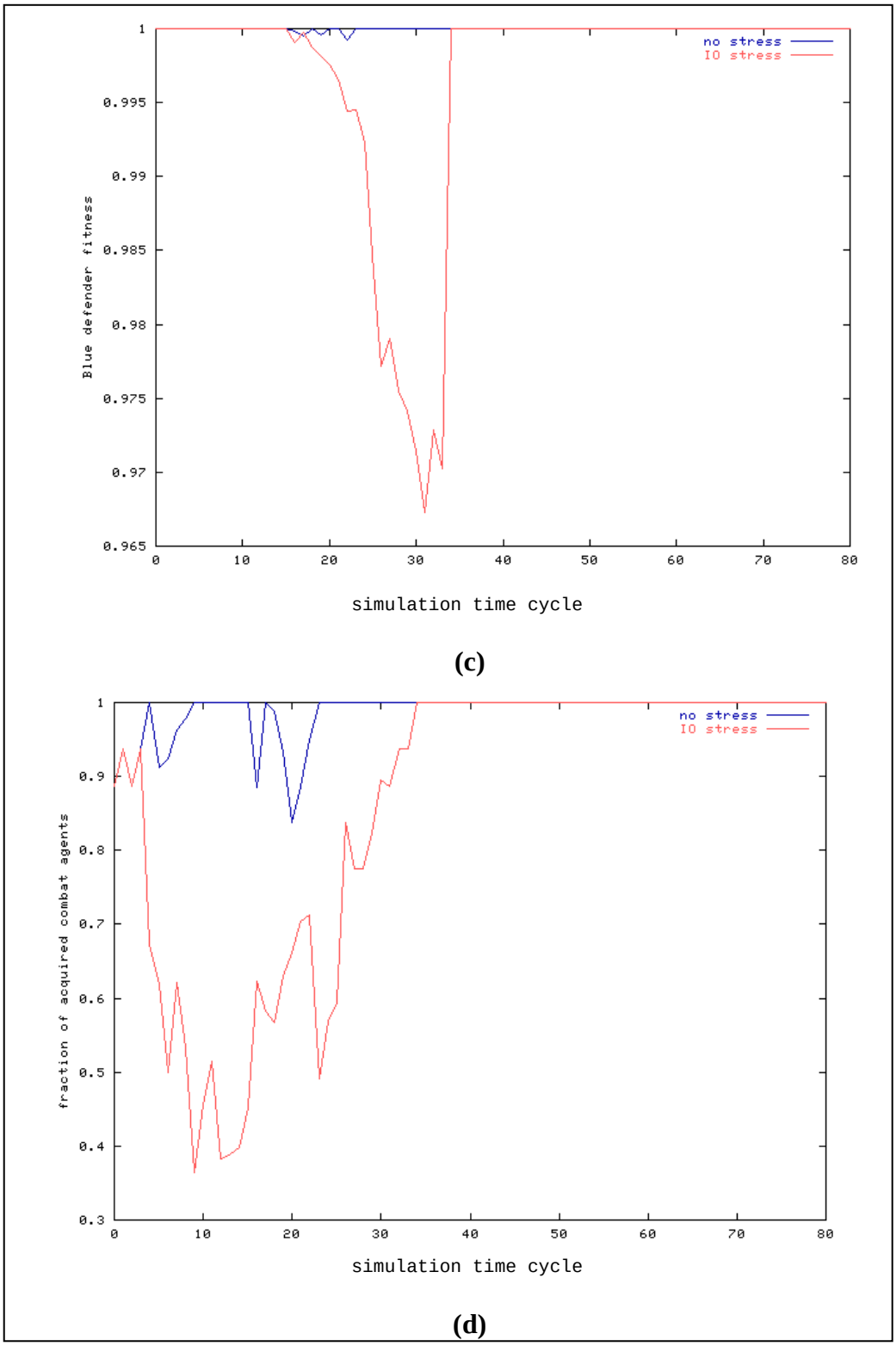


Figure 44. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 17 cells, and 1 Blue shot/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

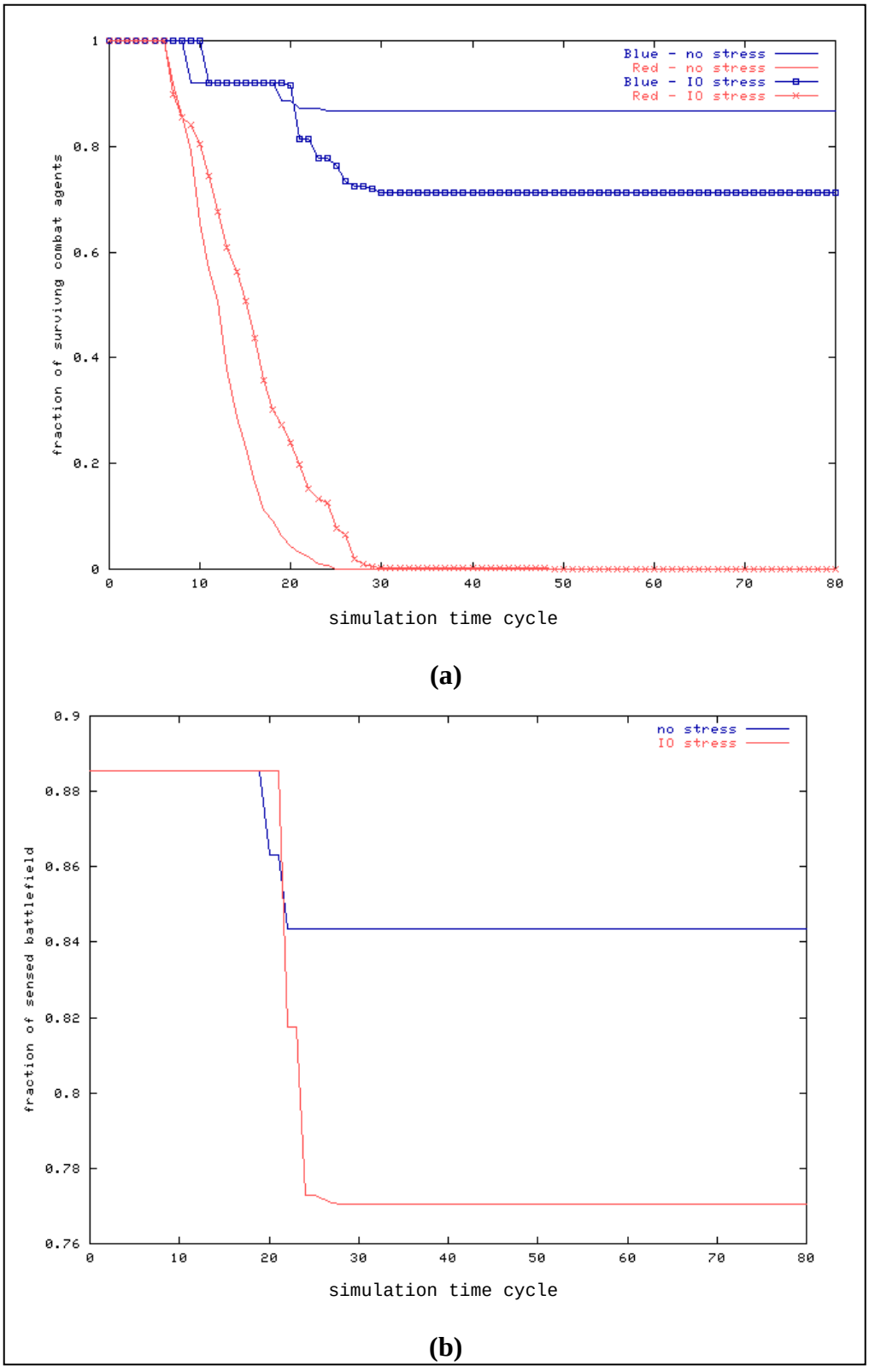


Figure 45. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 10 cells, and 3 Blue shots/cycle: (a) combat unit survival and (b) Blue sensor coverage.

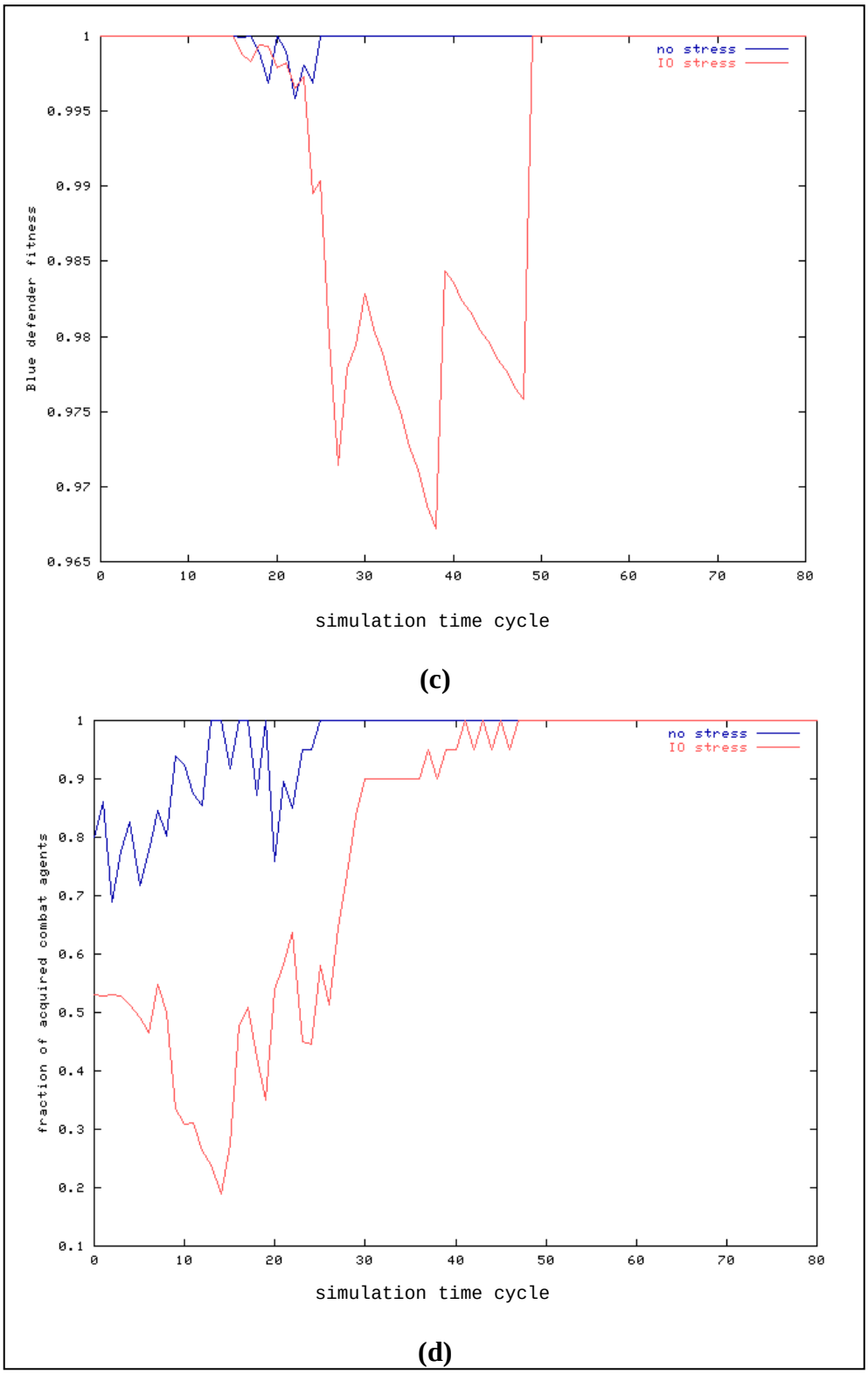


Figure 46. Fombler's Ford simulation results, with UGS density = 0.75, Blue fire point = 10 cells, and 3 Blue shots/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

defender fitness (Figure 46c), and Blue situational awareness of Red (Figure 46d) is actually inferior to the previous scenario variant (where Blue fire point = 17 cells). Red's average operational performance, on the other hand, is improved within this variant compared to that unit's performance in the previous scenario. Comparison of Blue and Red unit performance within this scenario relative to the previous scenario suggests that allowing Blue an earlier opportunity to engage invasive Red agents is more operationally significant than increasing the former's firepower engagement rate.

In the fifth Fombler's Ford scenario variant, the packing density of Blue's UGS network is further increased to a maximal level (with other parameters reflecting values from the second scenario variant). Simulation results from this scenario (averaged over the standard set of random seeds) are depicted in Figures 47 and 48. Now, the model parameters are next set to (a) UGS density = 1.00, (b) Blue fire point = 10 cells, and (c) Blue shots/cycle = 1; again, sets of simulations are run with and without jammer bombs. In this scenario, both Blue and Red average post-combat survival (Figure 47a) reflect levels observed in the fourth scenario variant simulations within both benign and IO-stressed environments. Blue's post-combat fractional sensor coverage (Figure 47b) in both benign and stressed environments is slightly improved over levels seen in the previous scenario variant (as would be expected given the maximal UGS network density). Interestingly, Blue's average defender fitness (Figure 48c), although reflecting similar levels as in the previous scenario, is degraded under stress for a much shorter time interval than that observed in all previous scenario variants. This is likely attributable to improved Blue situational awareness of Red agents crossing over the UGS network (resulting from the maximal UGS packing density). This is indeed confirmed in the plot depicting Blue's situational awareness of Red (Figure 48d), where IO-stressed Blue unit awareness is sustained at levels comparable to those seen in the third scenario variant (where fewer Red agents have the opportunity to cross the UGS network due to Blue's employment of an "early response" maximal fire engagement point).

In the sixth and last Fombler's Ford scenario variant, the values for Blue fire point, Blue shots/cycle, and UGS packing density used in the third, fourth, and fifth scenario variants, respectively, are combined to create an optimal Blue unit configuration (within the limits of the sensitivity analysis). Simulation results from this final scenario (averaged over the standard set of random seeds) are depicted in Figures 49 and 50. In this particular run, the model parameters are set to (a) UGS density = 1.00 (maximum density), (b) Blue fire point = 17 cells (i.e., Red agents are engaged as they are introduced at the edge of the CA lattice), and (c) Blue shots/cycle = 3 (with Red shots/cycle still equal to 1). Blue's average post-combat survival level (Figure 49a) is, in this case, only reduced by a little over 5% under IO stress (maintaining a new peak level of ~0.90), with an associated peak attrition rate of Red agents under these conditions. The average post-combat Blue sensor coverage (Figure 49b) is only reduced from ~0.932 to a little over 0.916, reflecting enhanced survivability of sensor-equipped Blue agents. Blue's defender fitness (Figure 50c) almost remains unchanged going from a benign to IO-stressed

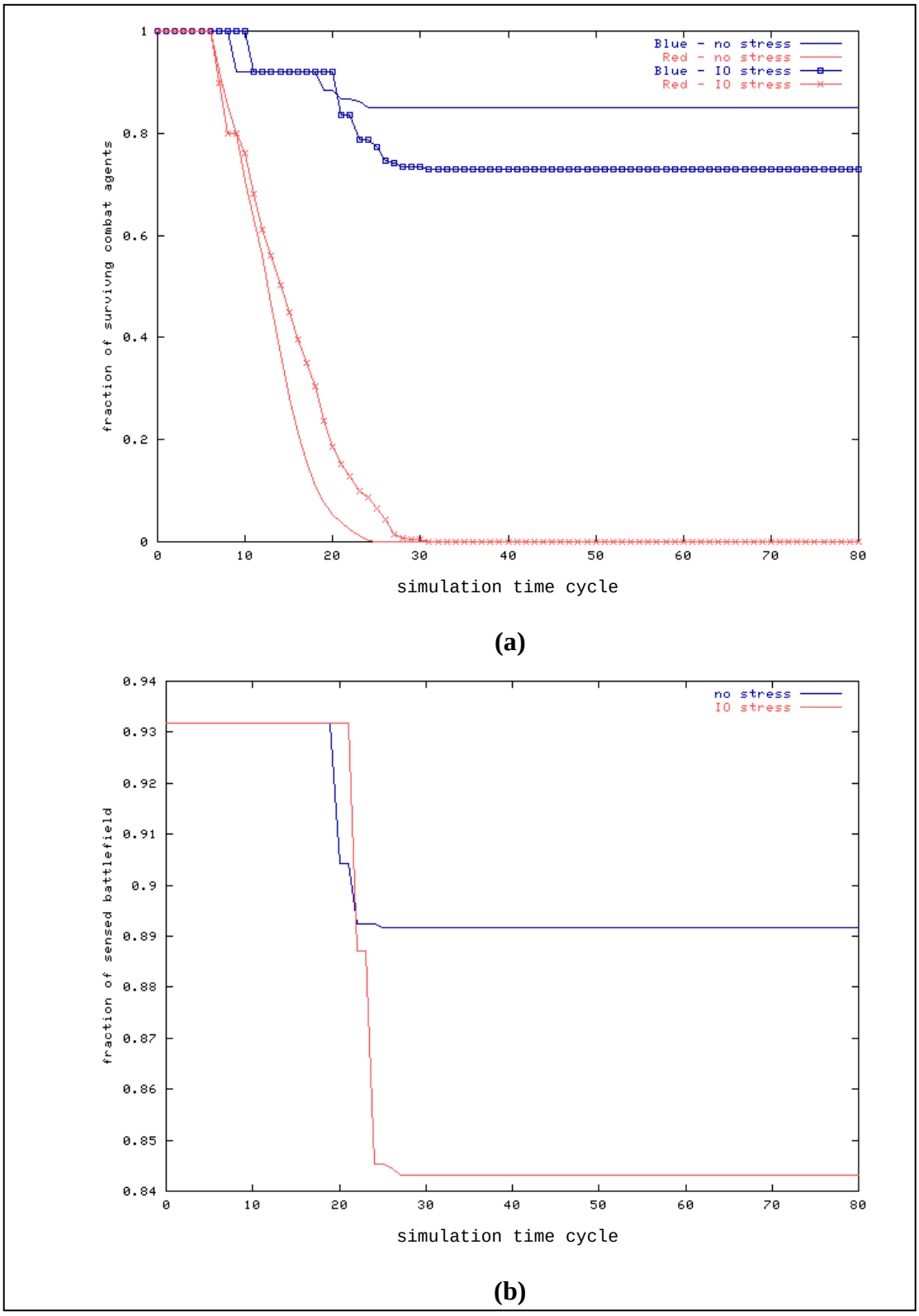


Figure 47. Fombler's Ford simulation results, with UGS density = 1.00, Blue fire point = 10 cells, and 1 Blue shot/cycle: (a) combat unit survival and (b) Blue sensor coverage.

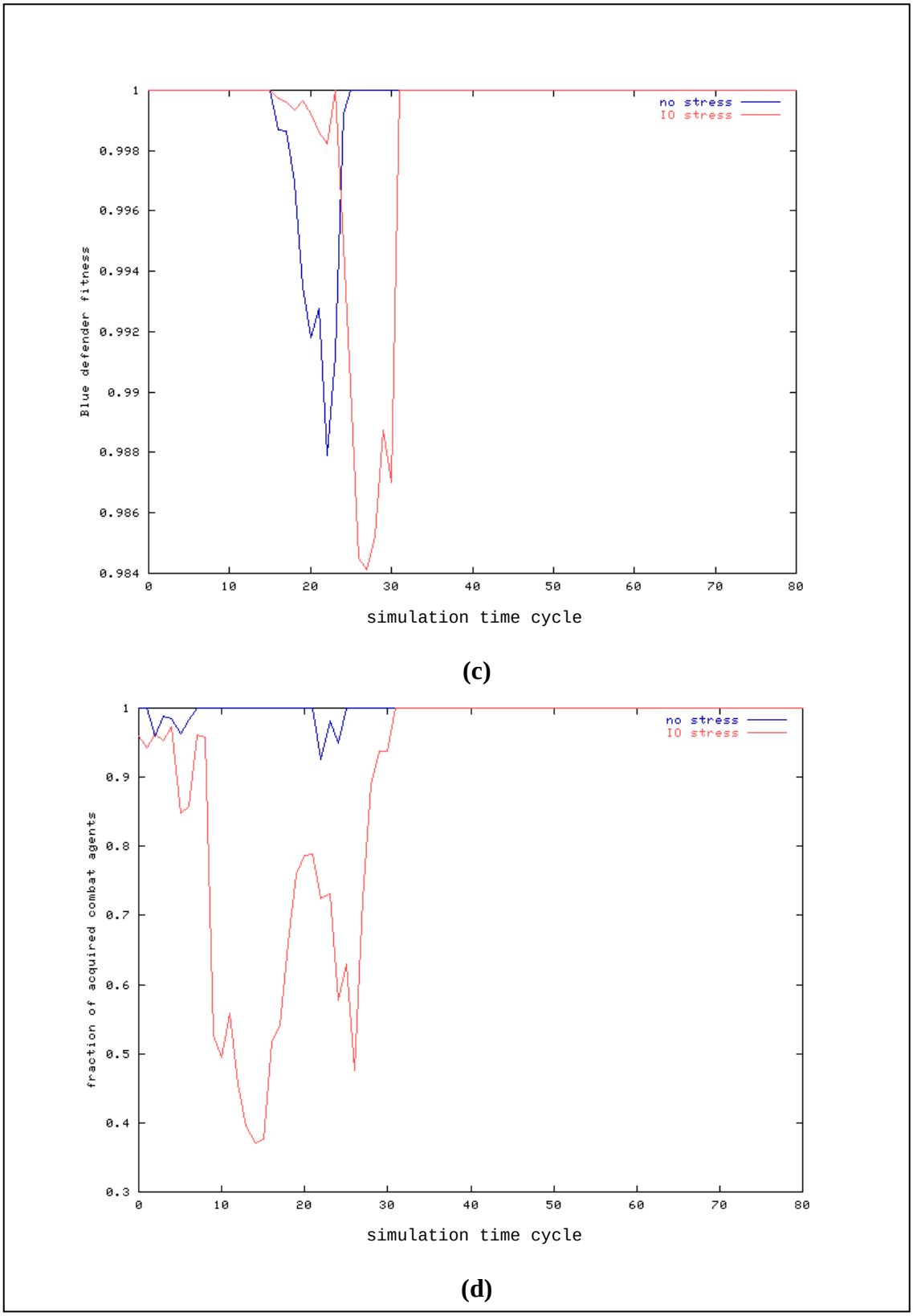


Figure 48. Fombler's Ford simulation results, with UGS density = 1.00, Blue fire point = 10 cells, and 1 Blue shot/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

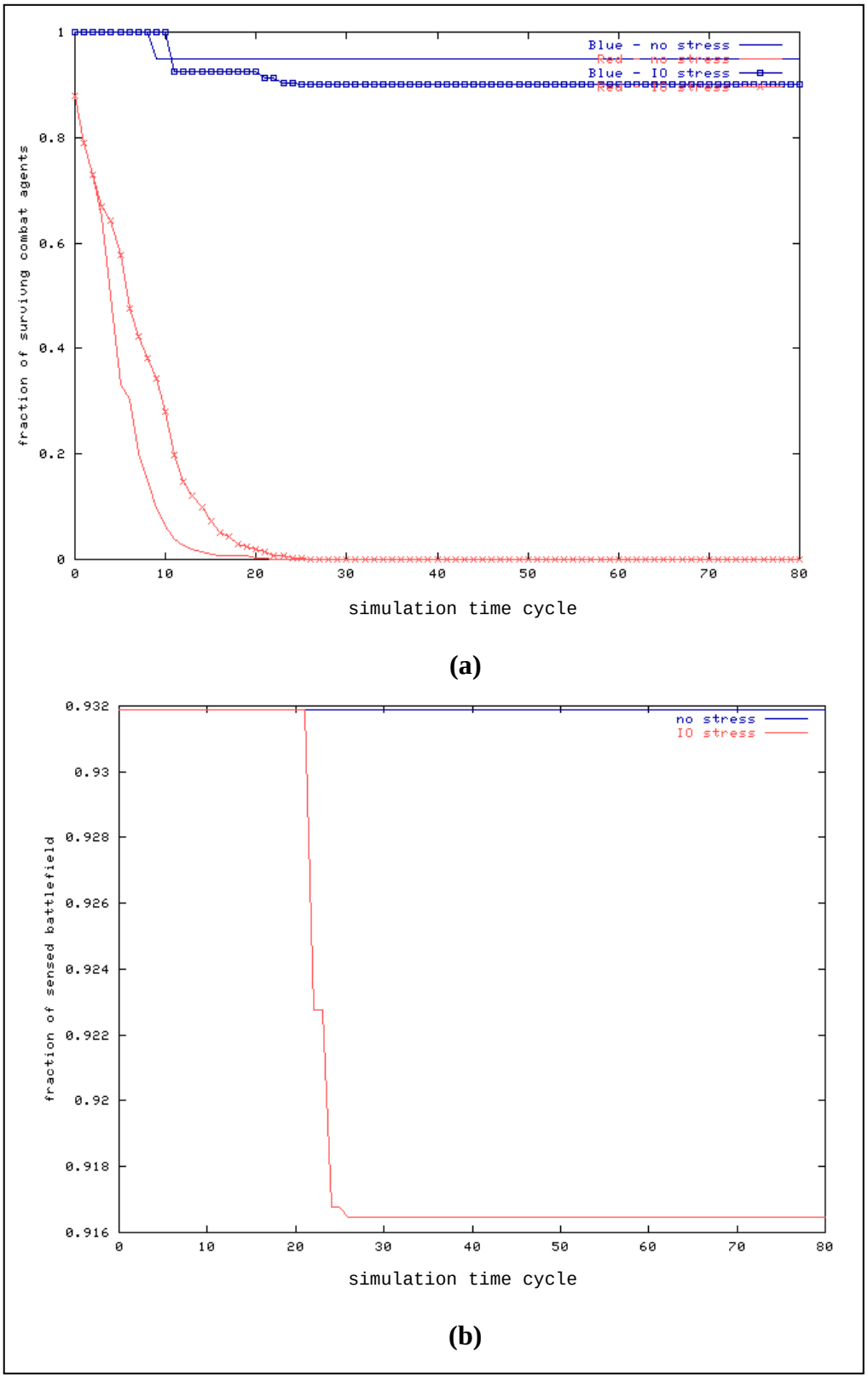


Figure 49. Fombler's Ford simulation results, with UGS density = 1.00, Blue fire point = 17 cells, and 3 Blue shots/cycle: (a) combat unit survival and (b) Blue sensor coverage.

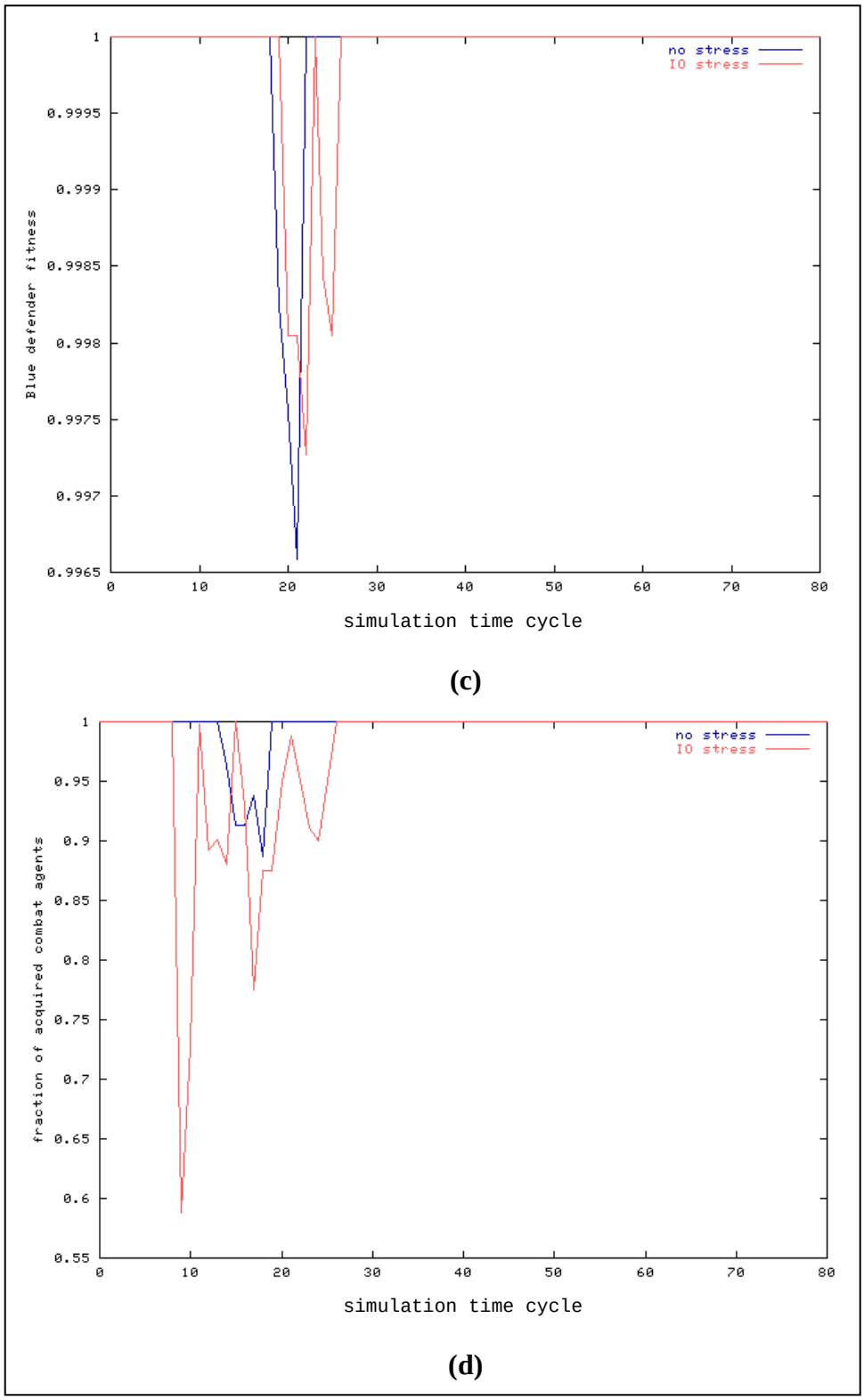


Figure 50. Fombler's Ford simulation results, with UGS density = 1.00, Blue fire point = 17 cells, and 3 Blue shots/cycle: (c) Blue's ability to prevent Red penetration into Blue territory and (d) Blue situational awareness of Red.

environment, while the Red agent sustains penetration into Blue territory reaching a minimal level (relative to model parameter constraints). Finally, Blue's situational awareness of Red (Figure 50d) is still degraded by jammer-bomb-induced UGS web discontinuities, but the sustained magnitude and duration of Blue's awareness here achieves peak levels relative to all other scenario variants studied under the sensitivity analysis. These plots quantitatively illustrate how providing the Blue unit with a combination of enhanced operational capabilities can act together to significantly attenuate the perturbative impact of a form of IO stress that results in denial of communication network connectivity.

Further insight into how enhanced Blue unit operational capabilities act to improve Blue's situational awareness of Red can be gained by characterizing the sustainment of the latter metric throughout a combat simulation. This is done through the use of a return map, which is (in the current context) a 2-D scatter plot of BSAR (i.e., Blue's situational awareness of Red) evaluated at simulation time cycle t_{n+1} vs. BSAR evaluated at time cycle t_n. A return map illustrates the dynamics of a one-dimensional time series (i.e., a time series with just one independent variable) in a recursive rather than successive format. Figures 51–53 illustrate BSAR return maps for the first, second, and sixth Fombler's Ford scenario variants, respectively (where each return map displays a superposition of data collected from the 20 simulation instances run for each variant). In Figure 51, the perpetuated BSAR levels (measured in an unstressed benign environment) resultant from the Blue UGS network deployed with a (subcritical) average packing density of 0.55 form a fairly dispersive point cluster with a CM located at 0.489 and 0.493.* This benign-environment BSAR cluster is seen to move towards the upper right-hand corner of the plot when the UGS packing density is increased to 0.75 (the blue square points in Figure 52), with the point cluster CM now located at 0.848 and 0.850. However, the introduction of IO stress into this scenario variant serves to slide the BSAR cluster back towards the lower left-hand corner (the red cross points in Figure 52), where the cluster CM degrades to a location of 0.555 and 0.554. Finally, as demonstrated in Figure 53, the implementation of maximal levels of UGS packing density, Blue fire point, and Blue shots per time cycle drives the benign-environment BSAR cluster (blue square points) far into the upper right-hand corner, with a point cluster CM located at 0.979 and 0.978. As previously stated (and is now clearly observable in this last return map), the introduction of IO stress into this last scenario variant has minimal impact on perpetuated BSAR levels, where the point cluster CM now only degrades to 0.953 and 0.952.

* All return map data are collected up to and including simulation time cycle 41 to reflect the maximal time window of nonzero Red combat agent survival levels observed across the Fombler's Ford scenario variants.

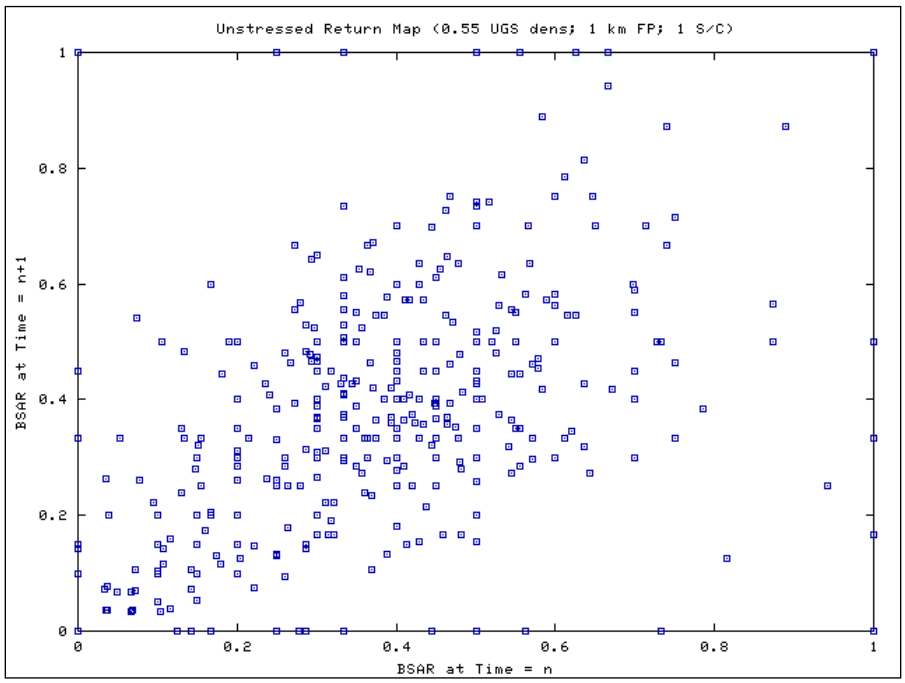


Figure 51. BSAR return map for the Fombler's Ford simulation, with UGS density = 0.55, Blue fire point = 10 cells, and 1 Blue shot/cycle.

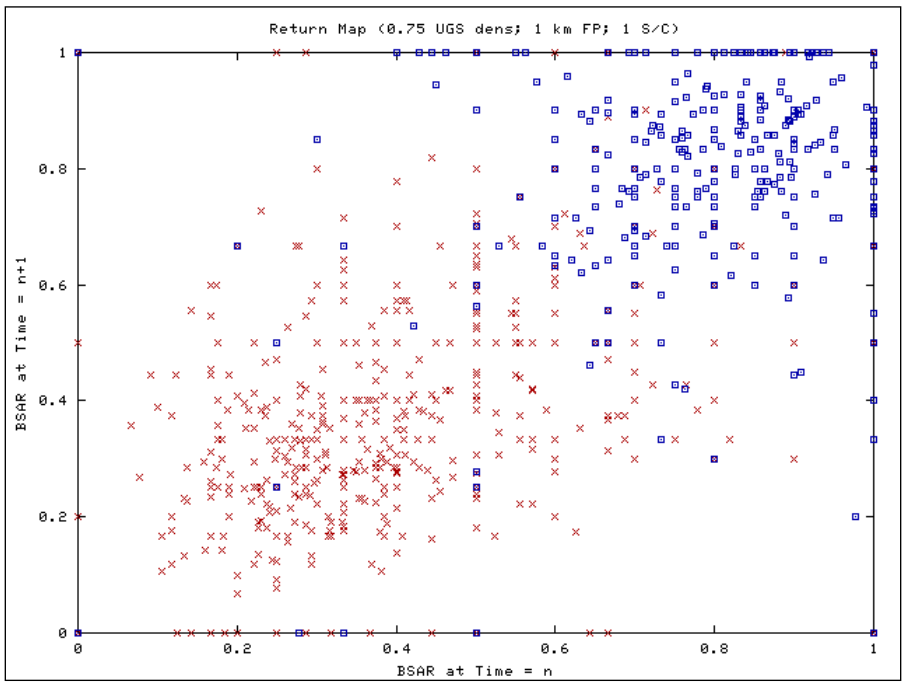


Figure 52. BSAR return map for both benign environment (blue squares) and IO-stressed (red crosses) Fombler's Ford simulations, with UGS density = 0.75, Blue fire point = 10 cells, and 1 Blue shot/cycle.

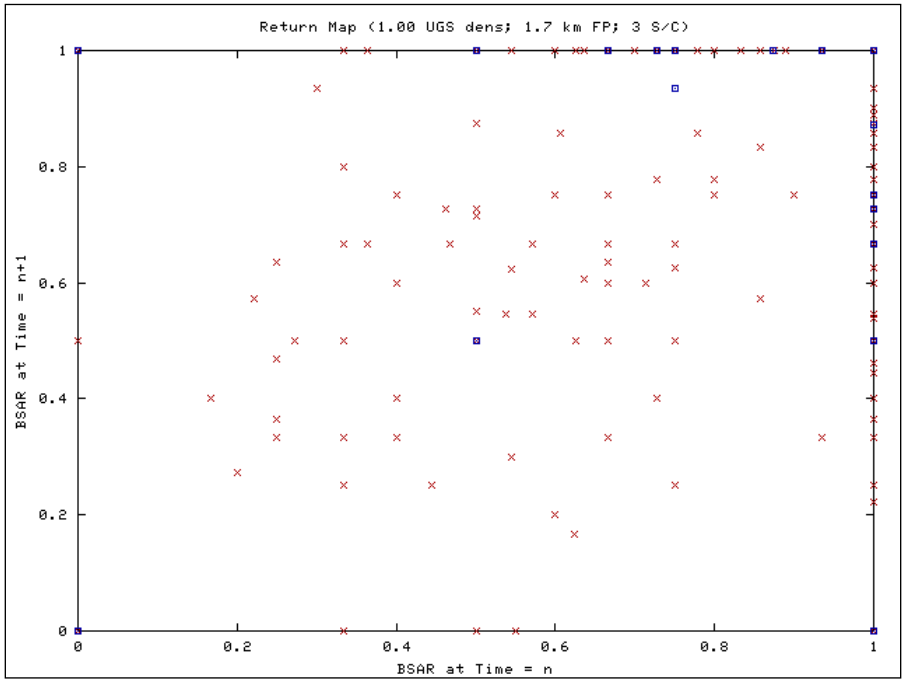


Figure 53. BSAR return map for both benign environment (blue squares) and IO-stressed (red crosses) Fombler's Ford simulations, with UGS density = 1.00, Blue fire point = 17 cells, and 3 Blue shots/cycle.

4. Conclusions

In this research, a CA-based software model of IO-stressed network-centric battle units has been developed and demonstrated. The resulting model serves to demonstrate both coevolutionary unit configuration optimization of sparring Blue/Red network-centric combat units (`combat_ga`/GA combination) and enhanced network-centric operations of a Blue combat unit in a defensive posture against Red invaders (`combat_proto`). In the former case, the nonconvergence of fitness levels is due (at least in part) to the limited network-centric functionality built into `combat_ga`, where Blue's access to greater communication ranges still doesn't guarantee that all Blue combat agents will always share the same collective situational "picture" of the battlespace. The combined CA/GA model does, however, demonstrate coevolutionary exploration of different combat agent classes based on hardware capability and behavioral configurations. In the case of the enhanced CA simulation engine, `combat_proto` serves to quantitatively demonstrate network-centric operations of a Blue combat unit in a defensive posture against Red invaders. In particular, the enhanced CA engine illustrates the operational impact of IO stress on Blue unit performance and the ways in which Blue can maintain operational robustness when encountering such stress.

Given that the combat agents used within the CA models are *reactive* in nature (with limited decision-making capability and no explicit "commander" type of agent able to formulate new tactical strategies when operationally required), the next step in model development is to provide agents with more deliberative behaviors, thus providing a framework for the hierarchical C2 decision-related structures ubiquitous to all military combat units. Current research into building more complex varieties of decision-making military command agents (Zhang et al. 2001) as well as hostile terrorist-style agents operating within densely-populated urban areas (Harper 2000) can provide a foundation for designing deliberative combat agents. Further work is still required, however, in order to realize rapidly adaptive agents operating within a network-centric combat structure.

5. References

Alberts, D. S., J. J. Gartska, and F. P. Stein. *Network-Centric Warfare: Developing and Leveraging Information Superiority.* Second edition (revised), DOD C4ISR Cooperative Research Program, pp. 115–131, February 2000.

Bar-Yam, Y. *Dynamics of Complex Systems.* Reading, MA: Perseus Books, pp. 546–549, 1997.

Eckart, J. D. "A Cellular Automata Simulation System: Version 2.0." *ACM SIGPLAN Notices*, vol. 27, no. 8, August 1992a.

Eckart, J. D. "Cellang 2.0: Language Reference Manual." *ACM SIGPLAN Notices*, vol. 27, no. 8, August 1992b.

Followill, F. E., J. K. Wolford, Jr., and J. V. Candy. "Advanced Array Techniques for Unattended Ground Sensor Applications." *Conference on Peace and Wartime Applications and Technical Issues for Unattended Ground Sensors, SPIE Proceedings Vol. 3081*, edited by G. Yonas, Orlando, FL, pp. 266–280, 21–25 April 1997.

Funes, P., and J. B. Pollack. "Measuring Progress in Coevolutionary Competition." *From Animals to Animats 6: Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, edited by J. Meyer et al., Cambridge, MA: MIT Press, 2000.

Gorman, GEN (Ret) P. *The Defense of Fombler's Ford: A Few Experiences in the Field Defense for Detached Posts That May Prove Useful in Our Next War*. Defense Advanced Research Projects Agency, Arlington, VA, 2000.

Haider, E. D. "Unattended Ground Sensors and Precision Engagement." Master's thesis, Naval Postgraduate School, Monterey, CA, September 1998.

Harper, K. A., S. S. Ho, and G. L. Zacharias. "Intelligent Hostile Urban Threat Agents for MOUT Operations." Proceedings of the 9th Conference on Computer Generated Forces and Behavior Representation, Orlando, FL, 2000.

Hencke, R. B. "An Agent-Based Approach to Analyzing Information and Coordination in Combat." Master's thesis, Naval Postgraduate School, Monterey, CA, September 1998.

Herman, M., and R. Hayes. "Measuring the Effects of Network-Centric Warfare." Prepared for the Office of the Secretary of Defence, Net Assessment, Contract DASW01-94-D-0043, Booz•Allen & Hamilton, pp. v–vi, April 1999.

Hillis, D. P, and R. P. Winkler. "The Fox and the Hare: a Coevolutionary Approach to Course of Action Generation." Advanced Displays and Interactive Displays Consortium Proceedings of the 5th Annual FedLab Symposium, College Park, MD, 2000.

Holland, J. H. *Hidden Order: How Adaptation Builds Complexity*. Reading, MA: Perseus Books, pp. 10–37, 1995.

Ilachinski, A. "Land Warfare and Complexity, Part I: Mathematical Background and Technical Sourcebook." Report no. CIM-461, Center for Naval Analyses, Alexandria, VA, pp. 85–92, July 1996.

Ilachinski, A. "Irreducible Semiautonomous Adaptive Combat (ISAAC): An Artificial-Life Approach to Land Combat." *Military Operations Research*, vol. 5, no. 3, pp. 29–46, 2000.

Maes, P., (ed). *Designing Autonomous Agents: Theory and Practice From Biology to Engineering and Back*. Cambridge, MA: MIT Press, 1990.

Mitchell, M. "Genetic Algorithms." *1992 Lectures in Complex Systems*, edited by L. Nadel and D. Stein, Reading, MA: Addison-Wesley, pp. 3–87, 1993.

Reynolds, C. W. "Flocks, Herds, and Schools: A Distributed Behavioral Model." *Computer Graphics*, vol. 21, no. 4, pp. 25–34, July 1987.

U.S. Department of the Army. *Information Operations*. FM 100-6, Washington, DC, August 1996.

Van Valen, L. "A New Evolutionary Law." *Evolutionary Theory 1*, pp. 1–30, 1973.

von Neumann, J. *Theory of Self-Reproducing Automata.* Edited and completed by A. Banks, Urbana, IL: University of Illinois Press, 1966.

Whitfield, J. "Ecology: Neutrality Versus the Niche." *Nature*, no. 417, pp. 480–481, 30 May 2002.

Wolfram, S. "Twenty Problems in the Theory of Cellular Automata." *Physica Scripta*, vol. T9, pp. 170–183, 1985.

Woodaman, R. F. A. "Agent-Based Simulation of Military Operations Other Than War Small Unit Combat." Master's thesis, Naval Postgraduate School, Monterey, CA, September 2000.

Zhang, Y., L. He, K. Biggers, J. Yen, and T. R. Ioerger. "Simulating Teamwork and Information Flow in Tactical Operations Centers Using Multi-Agent Systems." *Proceedings of the 10th Conference on Computer Generated Forces*, pp. 529–539, 2001.

Ziff, R. "On the Spanning Probability in 2-D Percolation." *Physical Review Letters*, vol. 69, pp. 2670–2673, 1992.

zum Brunnen, R. L., C. D. McDonald, P. R. Stay, M. W. Starks, and A. L. Barnes. "Information Operations Vulnerability/Survivability Assessment (IOVSA): Process Structure." ARL-TR-2250, U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, pp. 19–21, June 2000.

Appendix. Blue/Red Genome Classification for the Coevolutionary Simulation

In this appendix, plots of the coevolutionary dynamics of both Blue and Red combat unit genomes across 1360 successive generations are presented in Figures A-1 through A-19. In these plots, the coevolutionary dynamics are expressed relative to genome membership in various sets of combat agent classes (Tables A-1 through A-19) based on one or more chromosome genes. In all, 19 different types of combat agent classes are defined (based on the 23 chromosome genes preconfigured as mutable in the Blue/Red coevolution example presented in section 3.1.6), with a table summarizing the agent classes relative to a gene (or combination of genes) and associated plot of population per class (partitioned into Blue and Red agent populations) as a function of coevolutionary generation. In the cases where gene values are partitoned into a set of classes defined by very low, low, moderate, high, and very high levels, allele ranges per class are proportional to the unit interval partition used by Guzie for vulnerability risk assessment.[1]

[1] Guzie, G. L. "Vulnerability Risk Assessment." ARL-TR-1045, U.S. Army Research Laboratory, White Sands Missile Range, NM, pp. 21–32, June 2000.

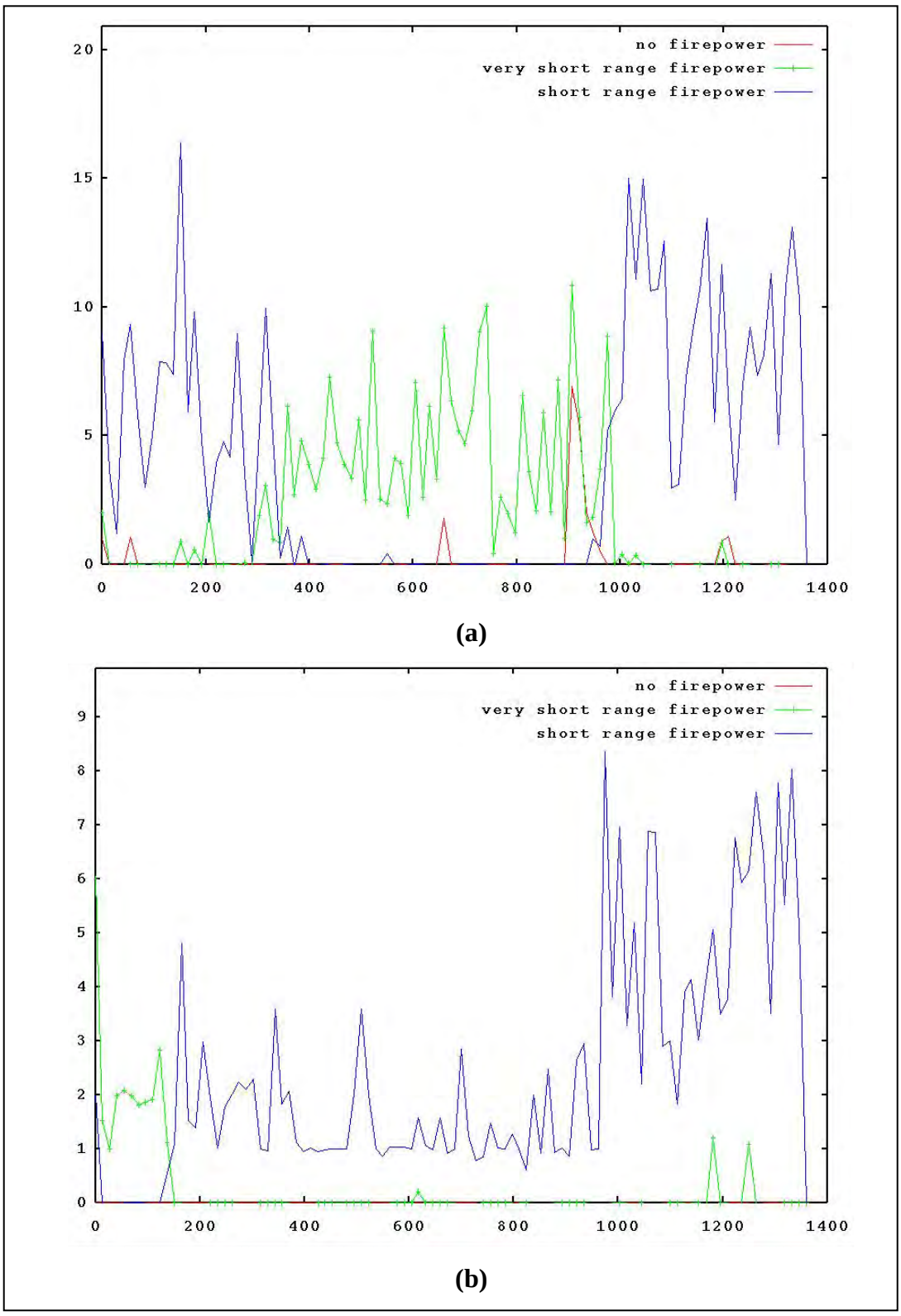


Figure A-1. Coevolutionary dynamics of firepower range classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

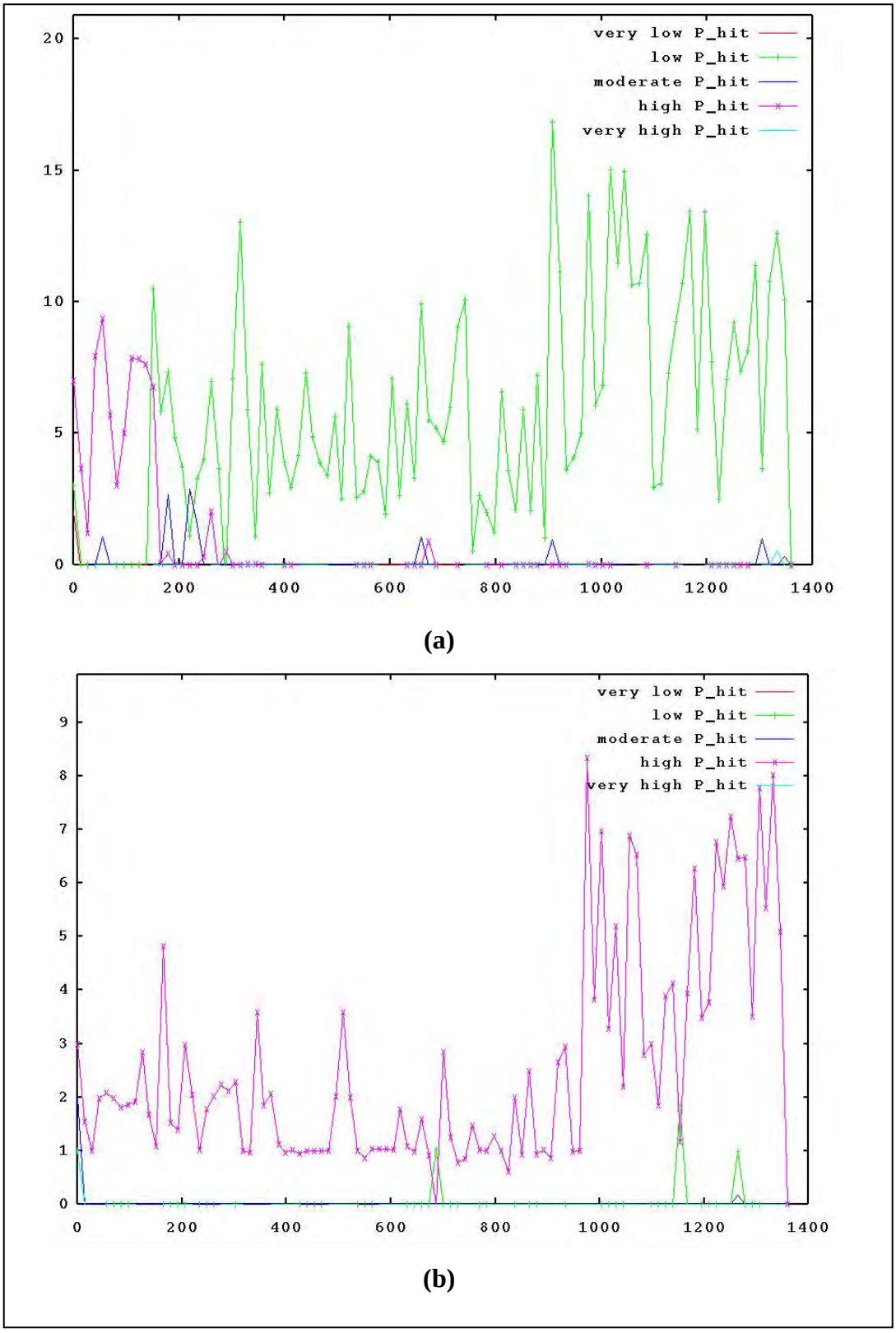


Figure A-2. Coevolutionary dynamics of single-shot P_{hit} classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

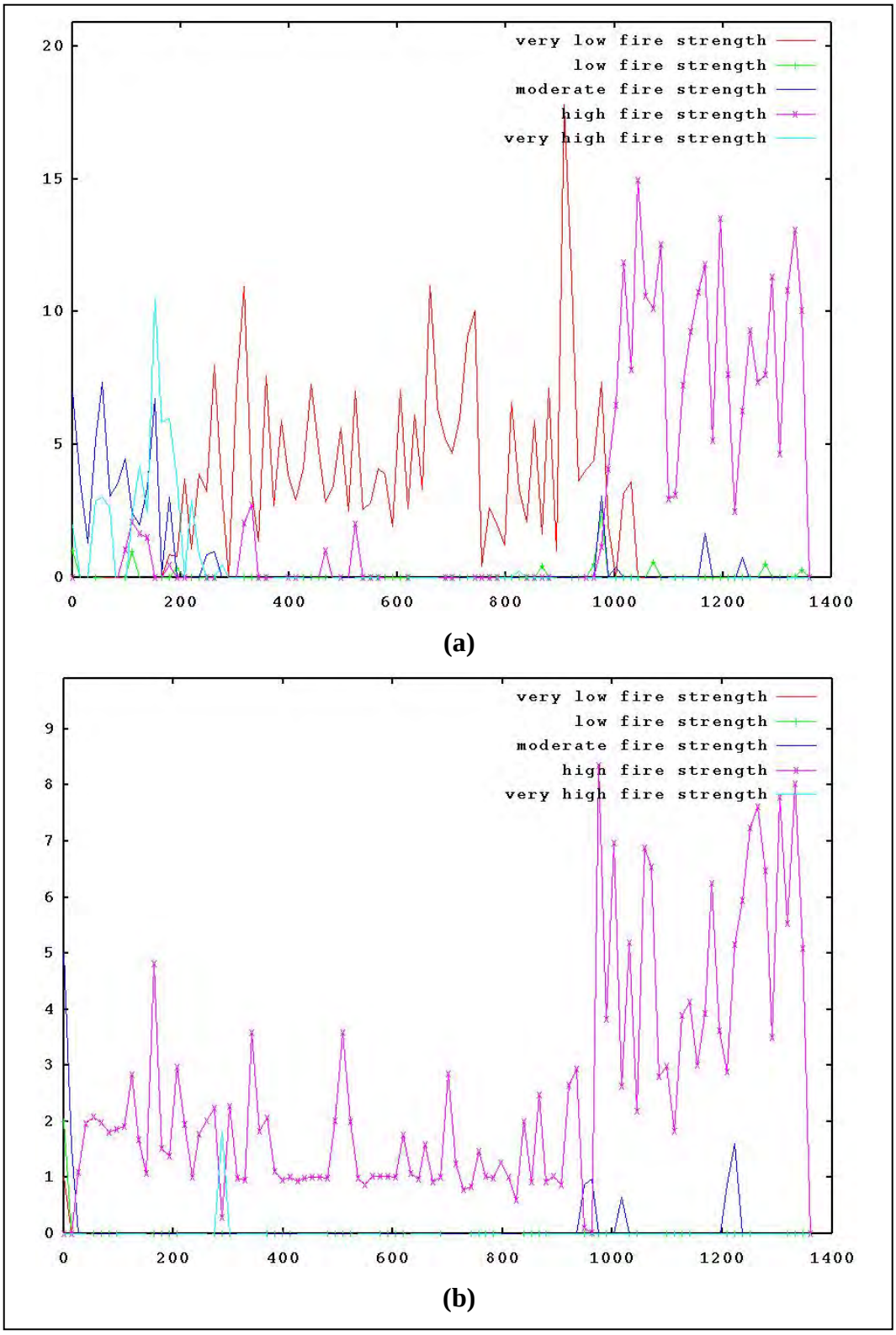


Figure A-3. Coevolutionary dynamics of firepower strength classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

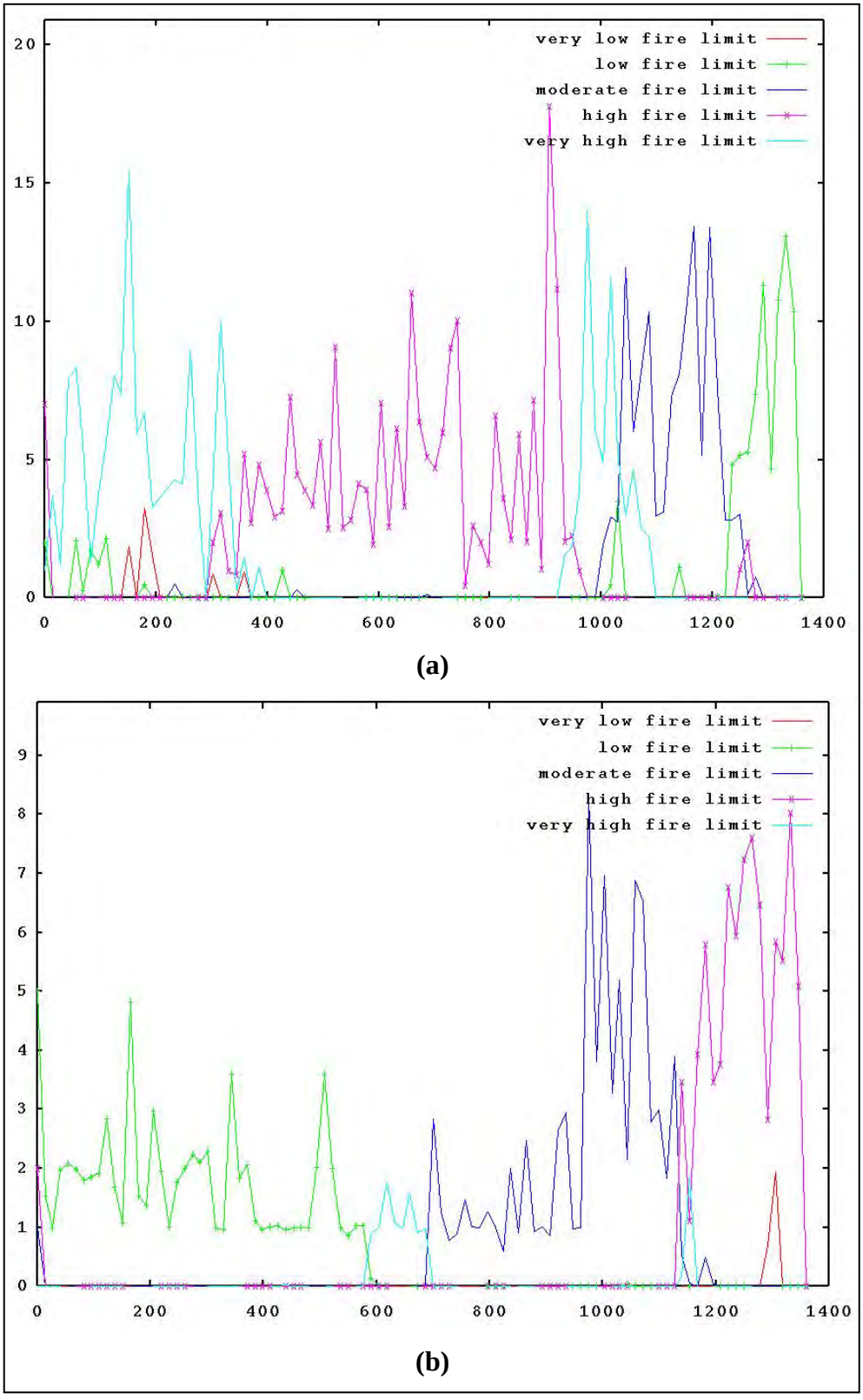


Figure A-4. Coevolutionary dynamics of firepower round limit classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

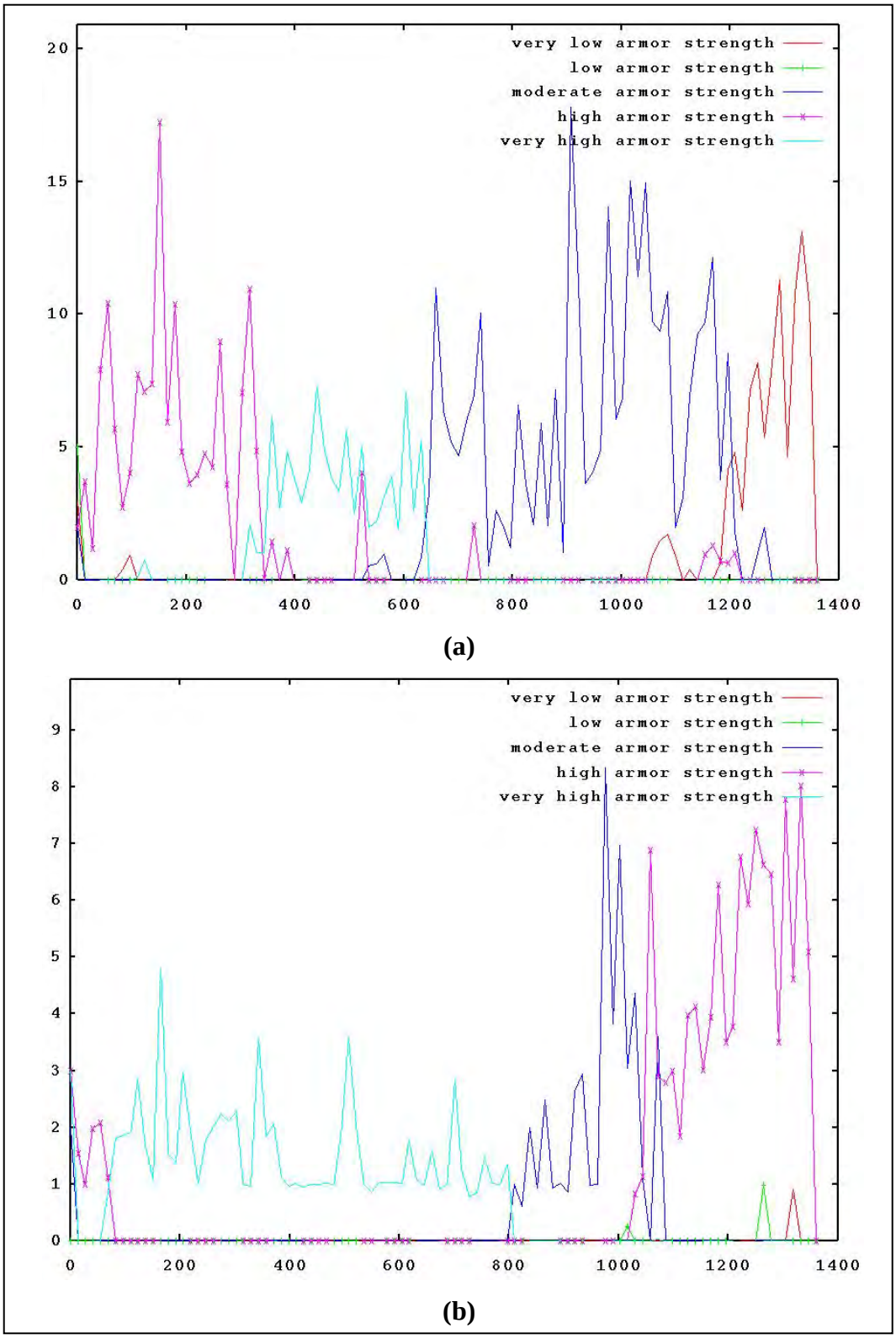


Figure A-5. Coevolutionary dynamics of armor strength classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

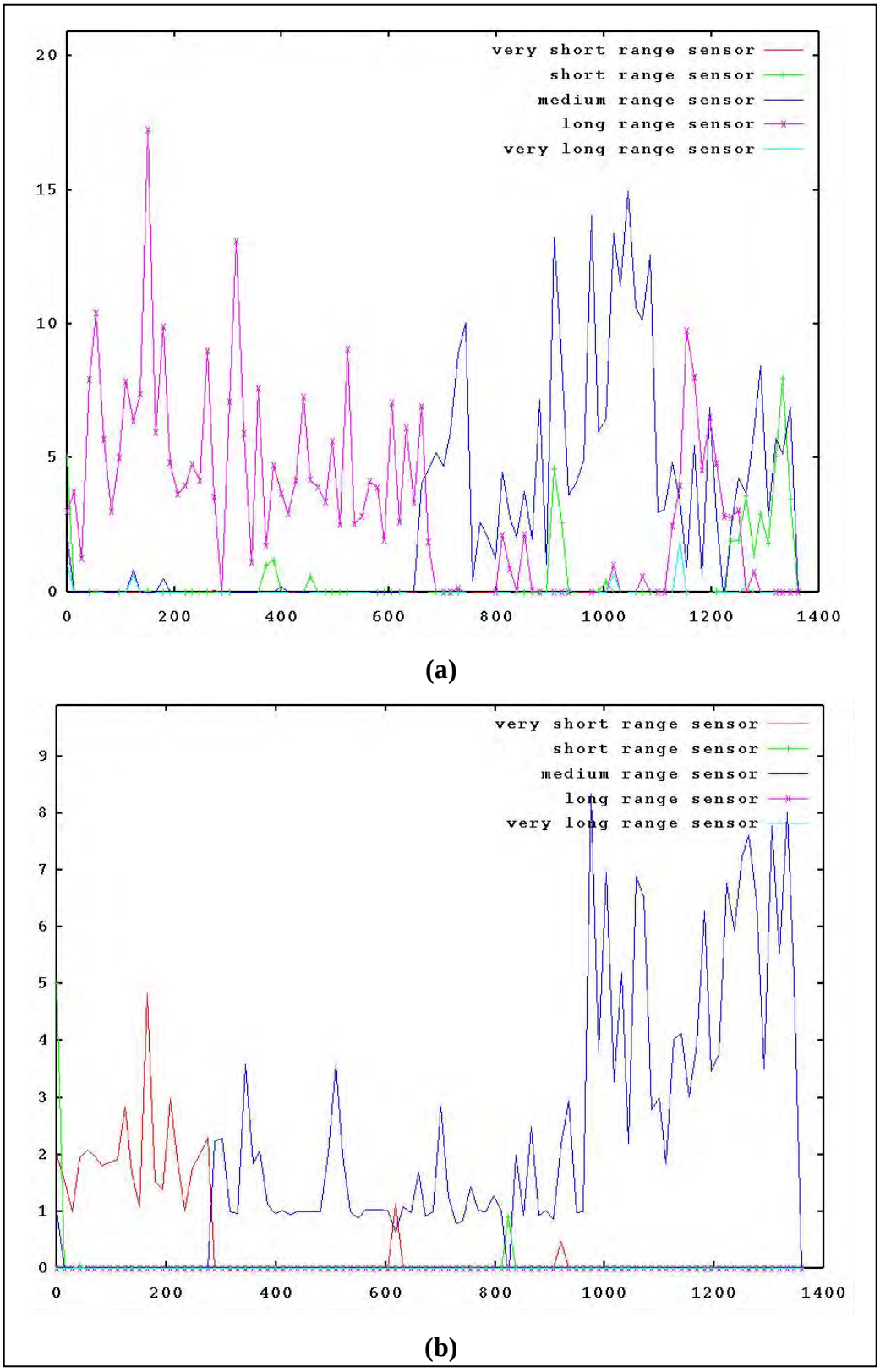


Figure A-6. Coevolutionary dynamics of sensor range classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

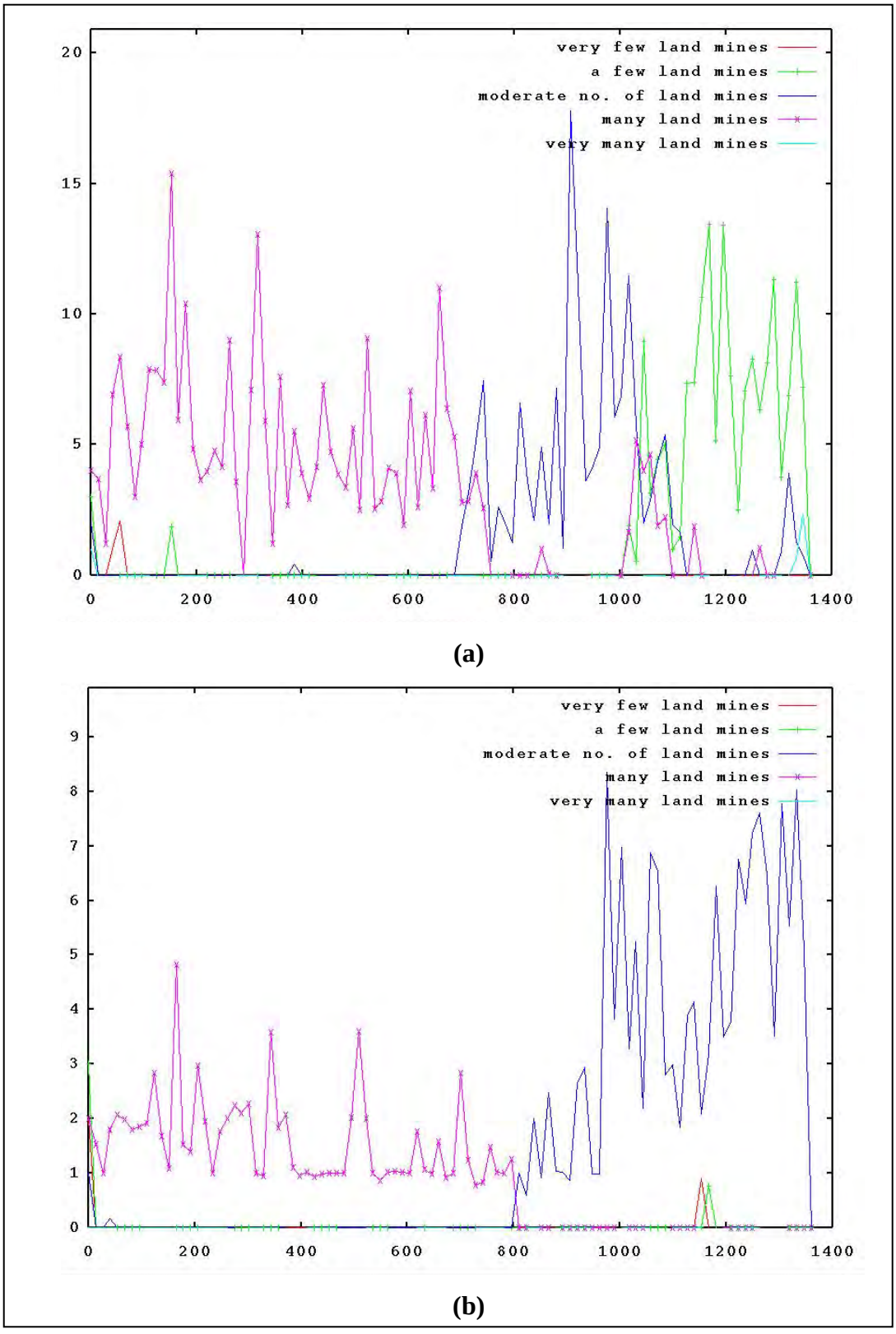


Figure A-7. Coevolutionary dynamics of carried land mine classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

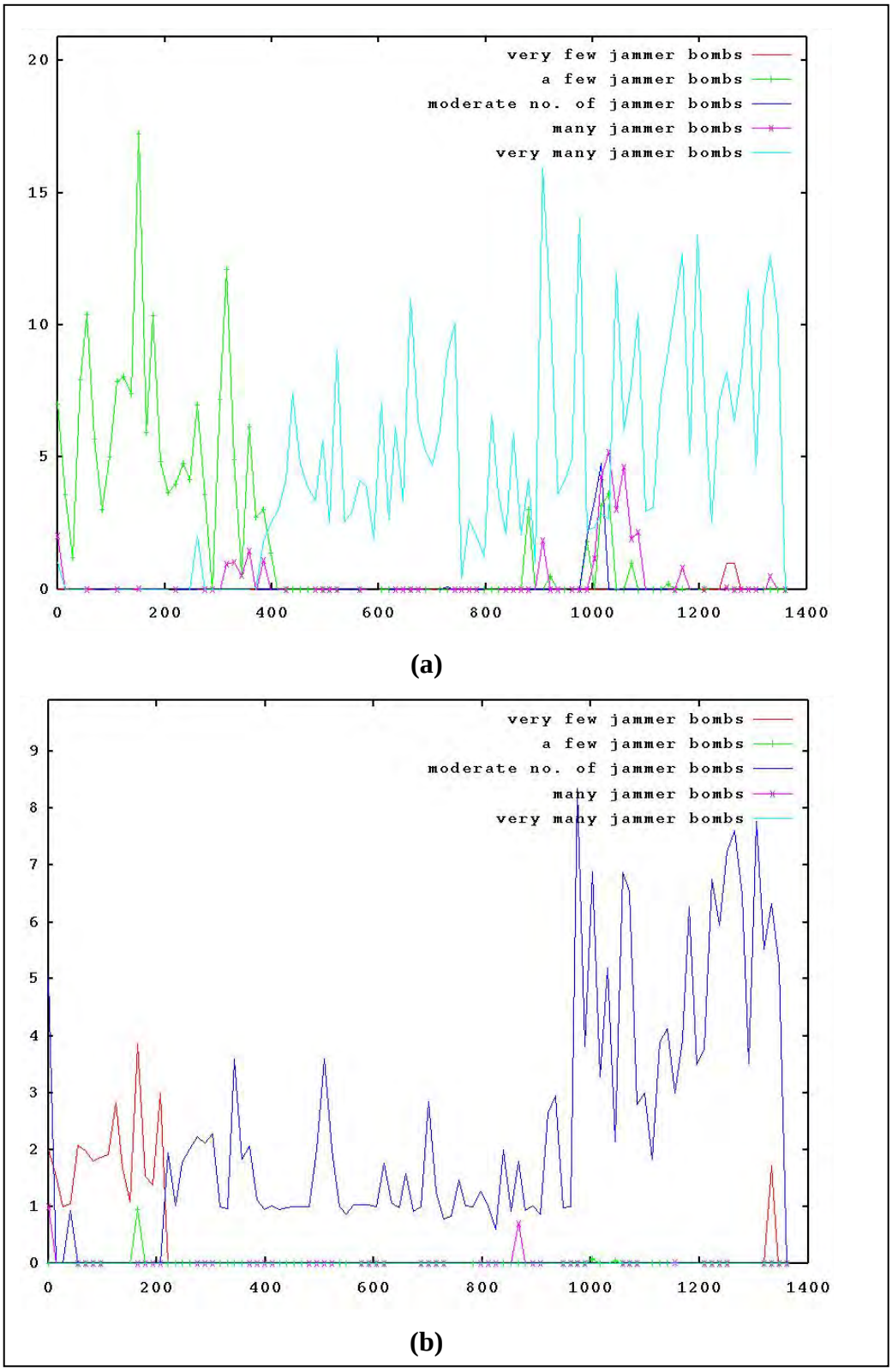


Figure A-8. Coevolutionary dynamics of carried RF jammer bomb classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

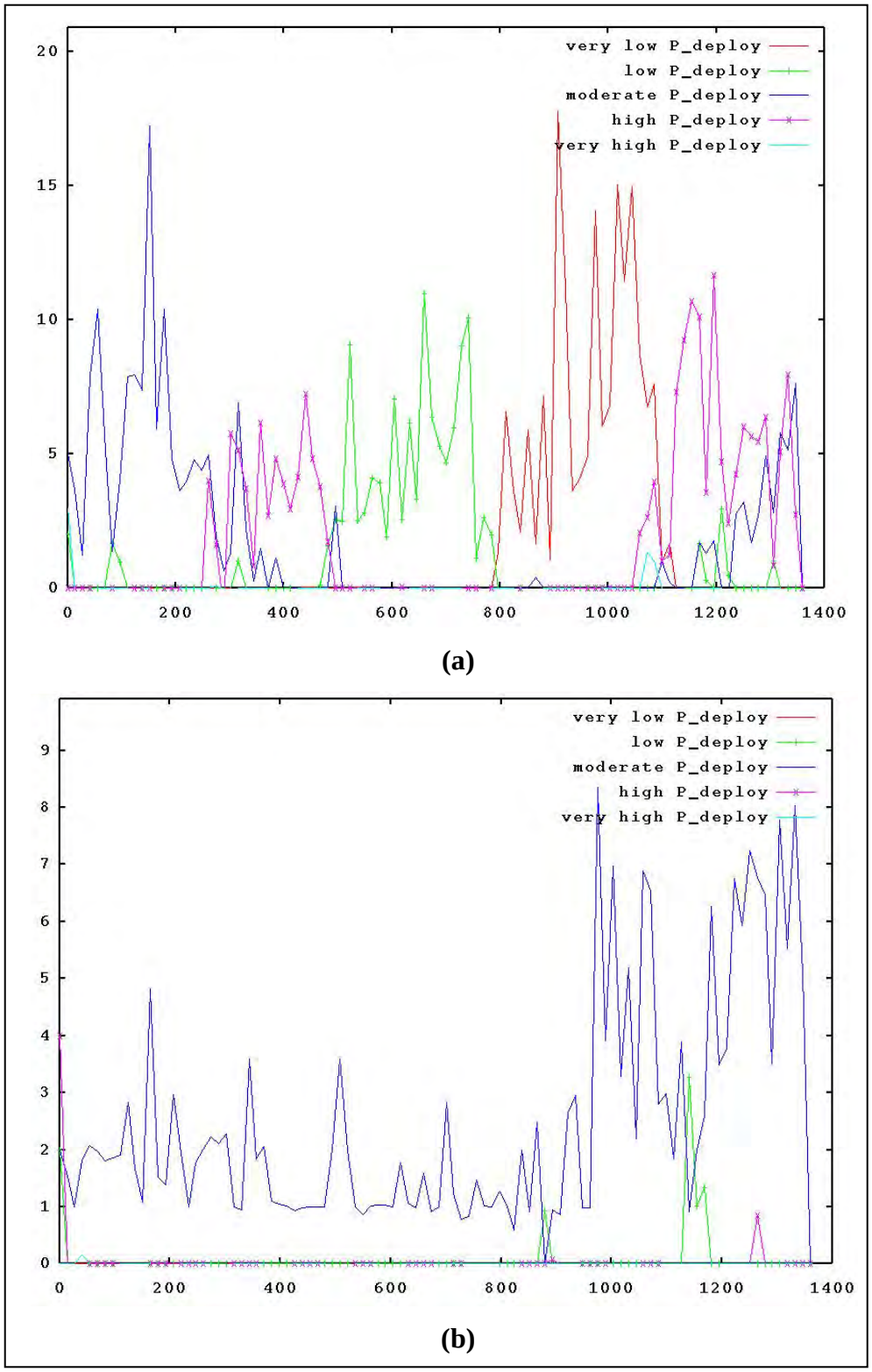


Figure A-9. Coevolutionary dynamics of mine/jammer bomb deployment probability with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

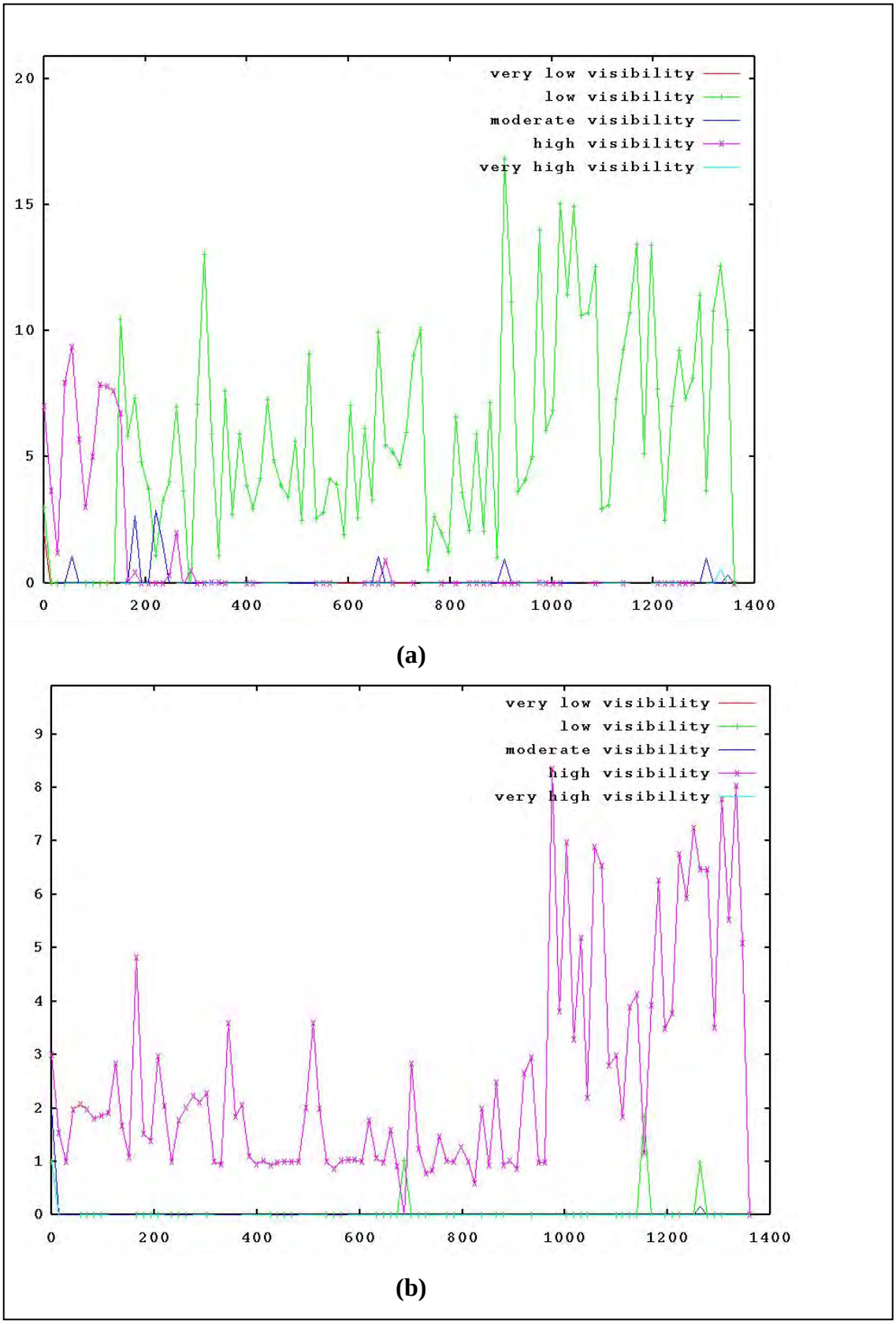


Figure A-10. Coevolutionary dynamics of signature classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

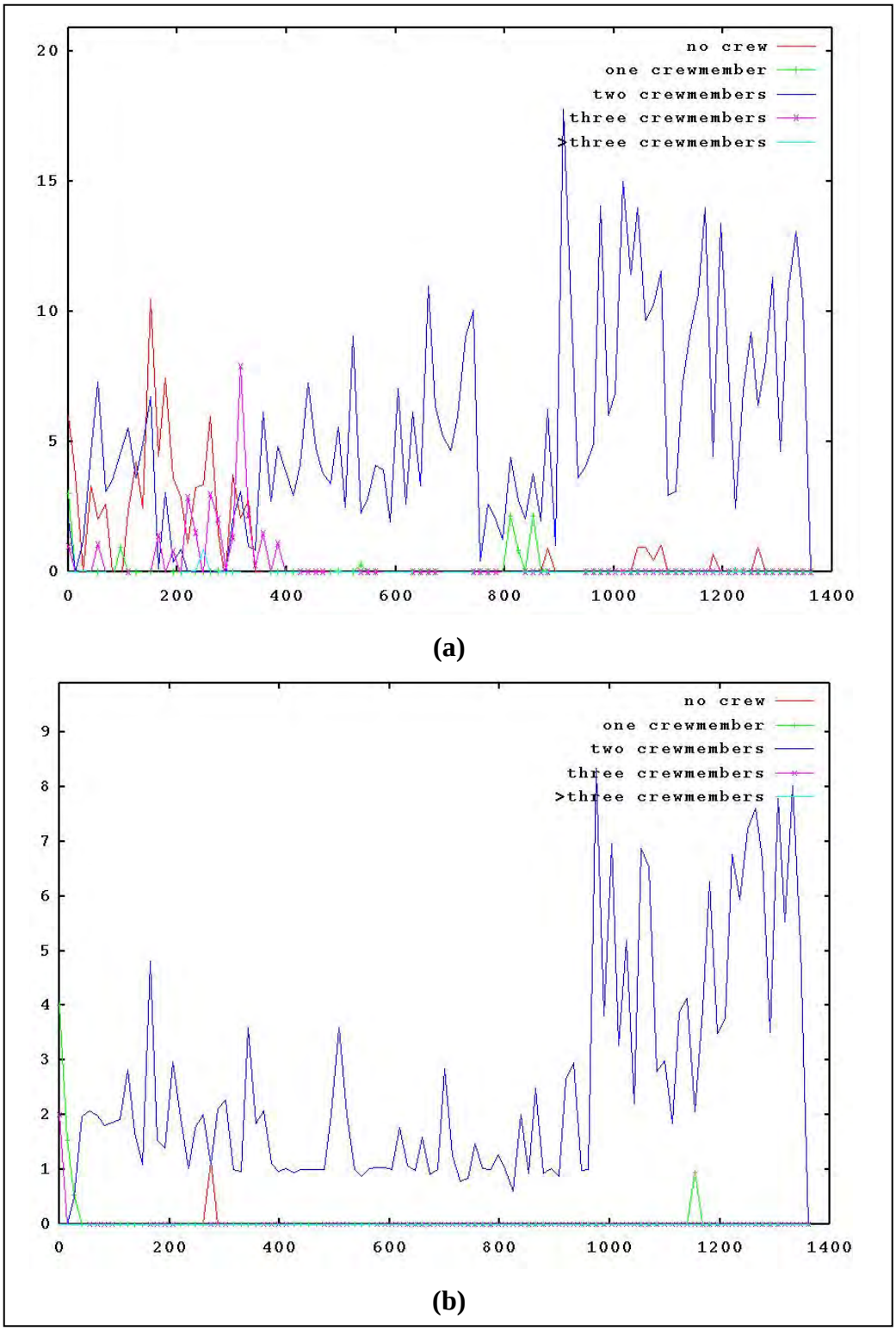


Figure A-11. Coevolutionary dynamics of crew classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

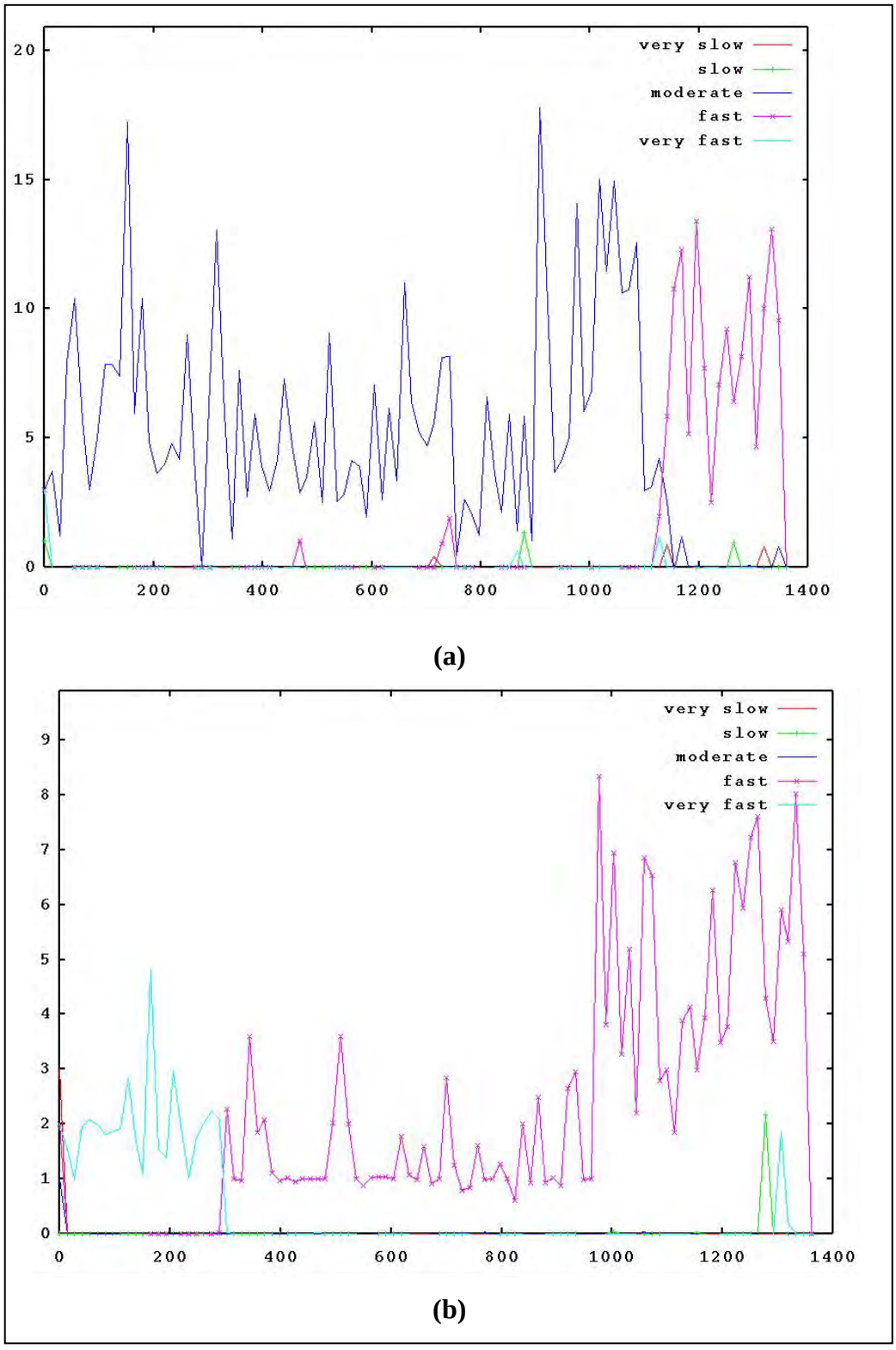


Figure A-12. Coevolutionary dynamics of maximum speed classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

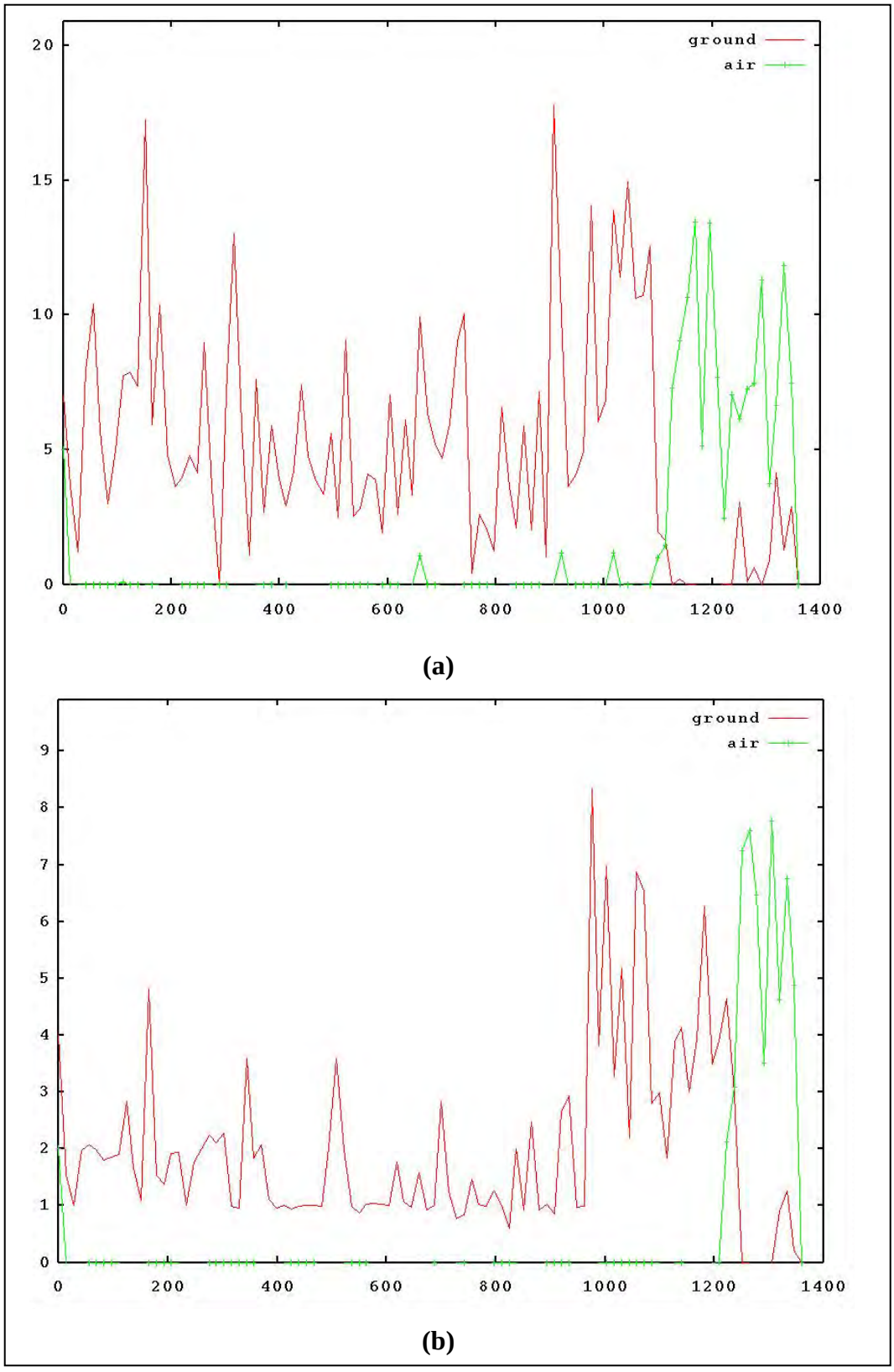


Figure A-13. Coevolutionary dynamics of transportation mode classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

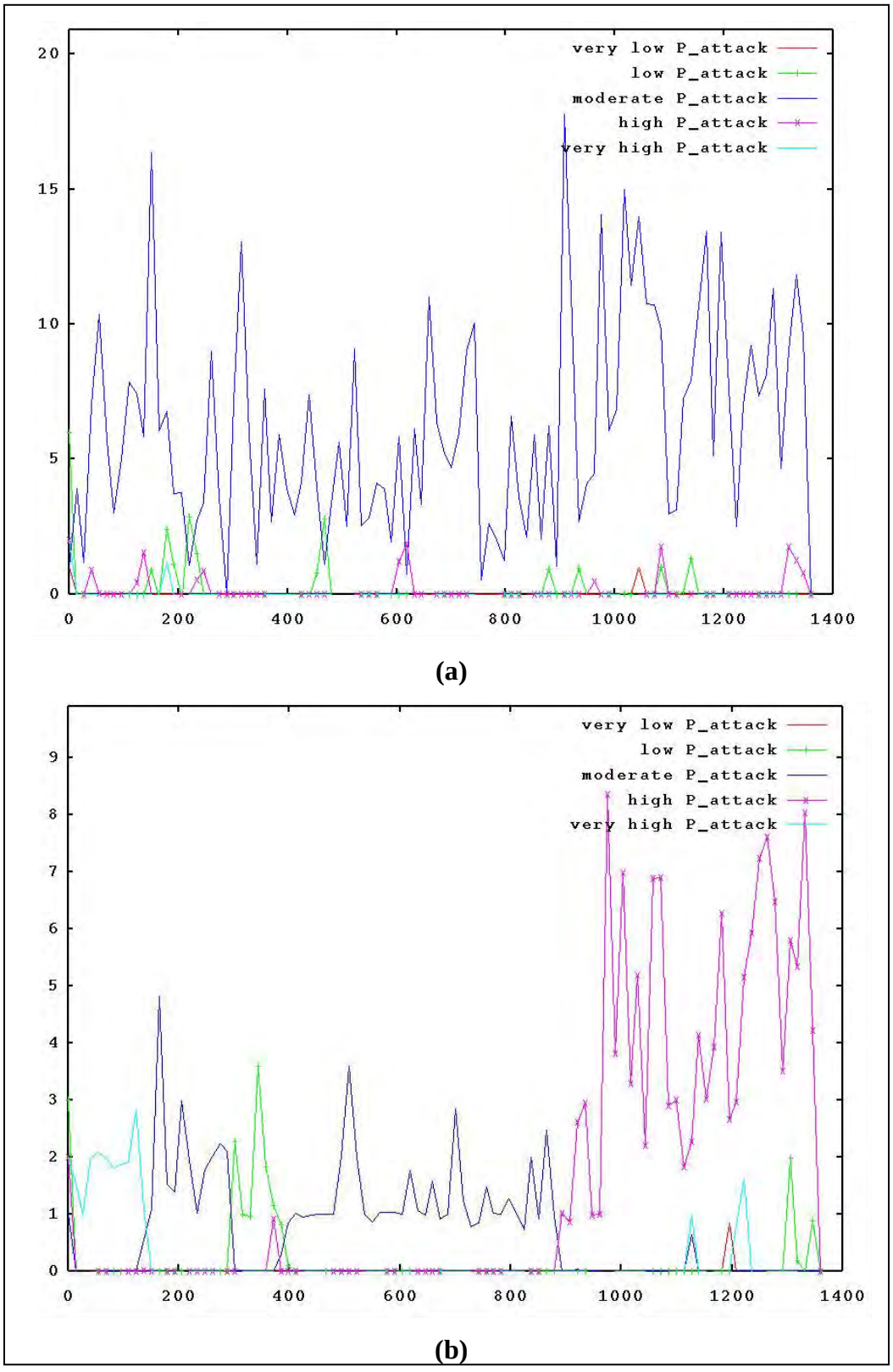


Figure A-14. Coevolutionary dynamics of attack initiation classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

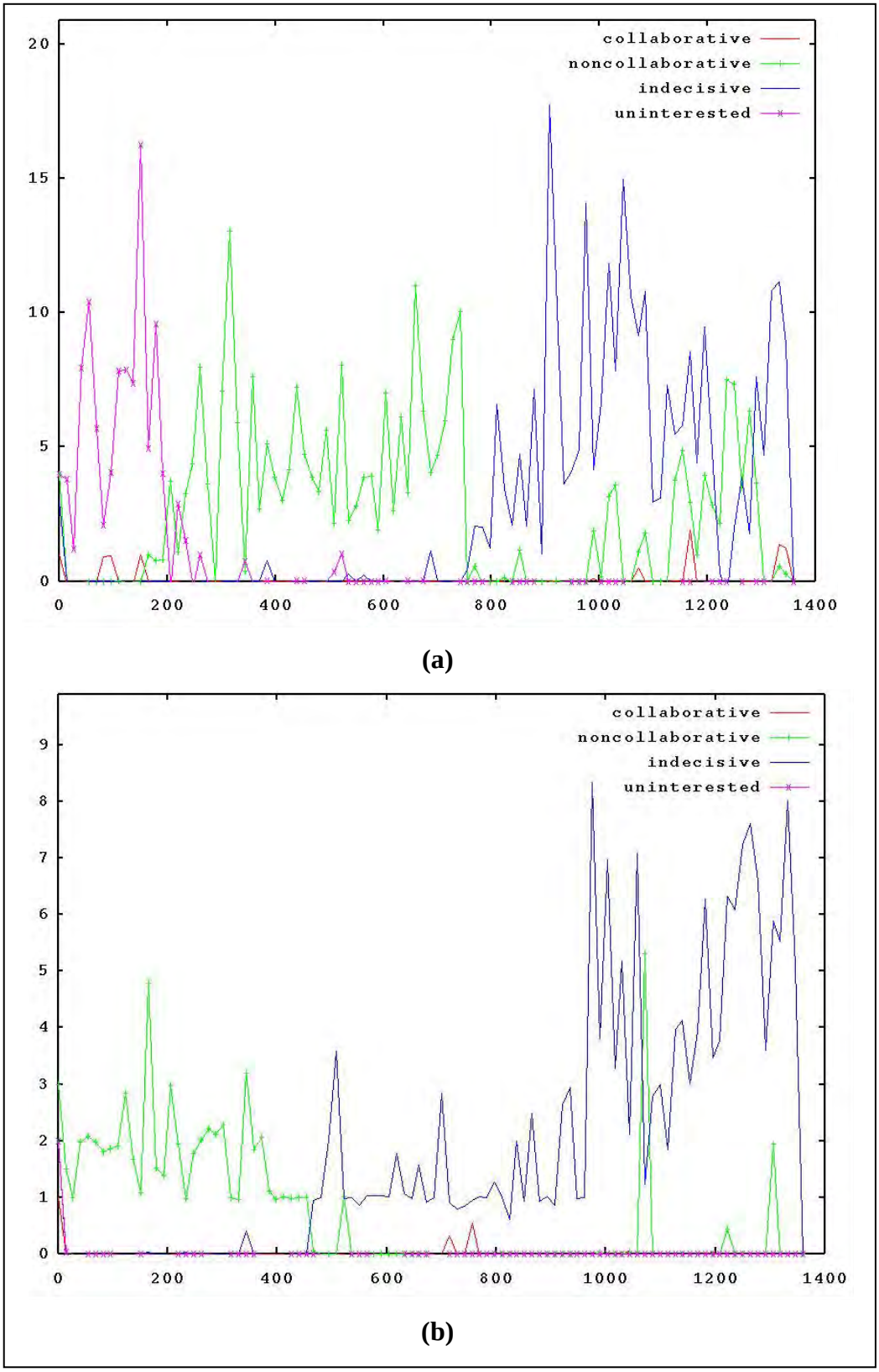


Figure A-15. Coevolutionary dynamics of friendly attraction/repulsion classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

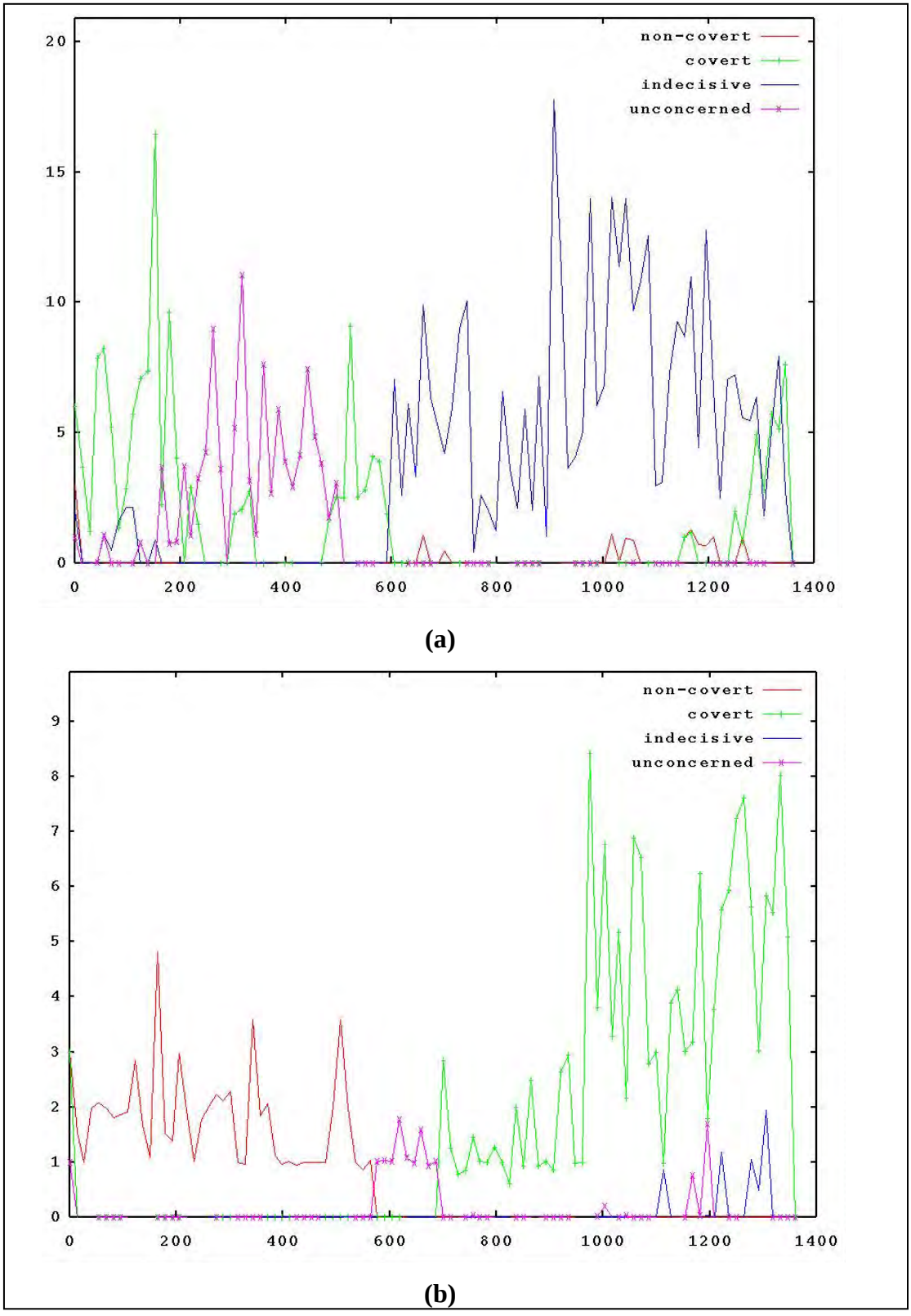


Figure A-16. Coevolutionary dynamics of enemy attraction/repulsion classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

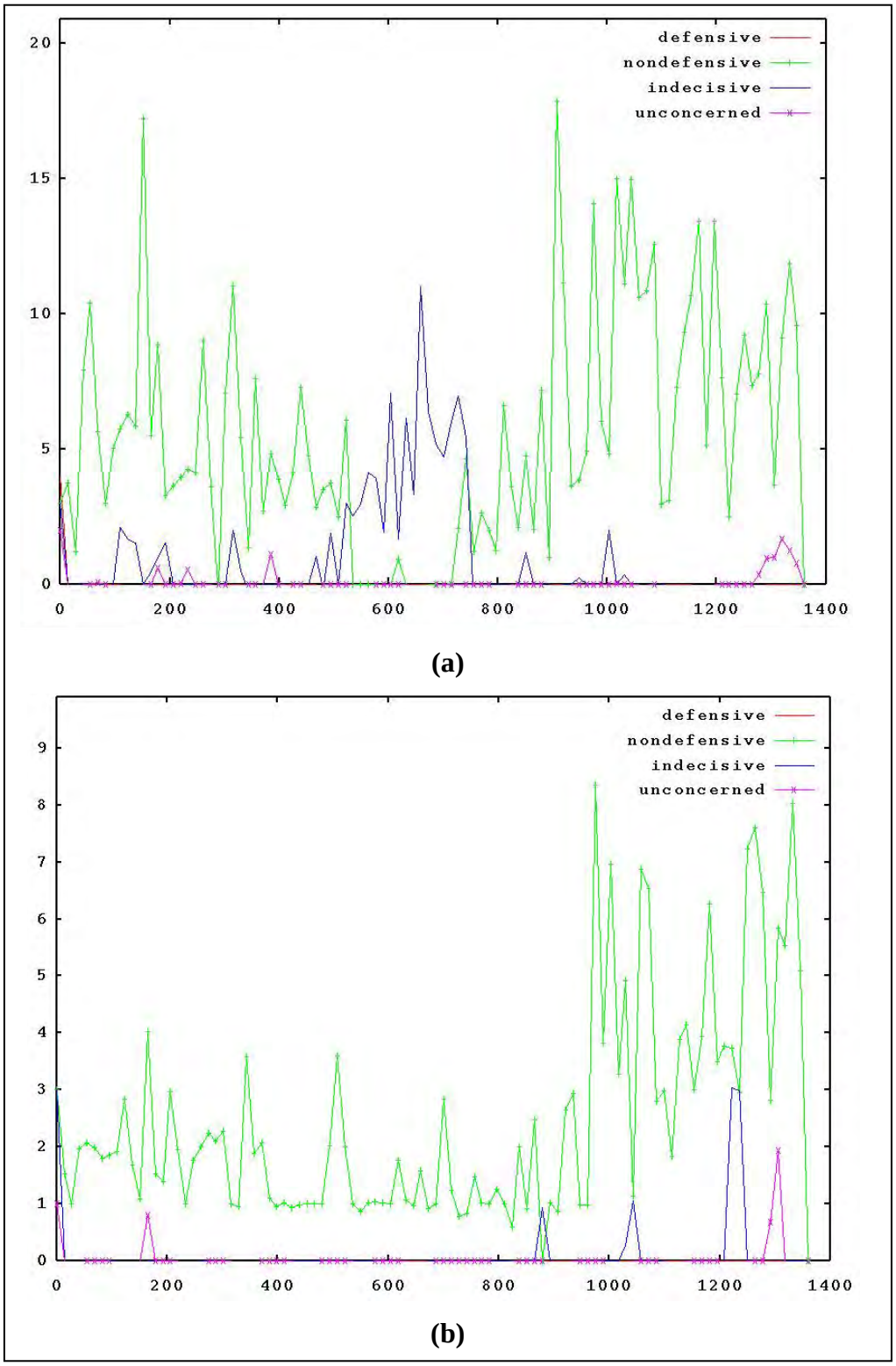


Figure A-17. Coevolutionary dynamics of friendly territory attraction/repulsion classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

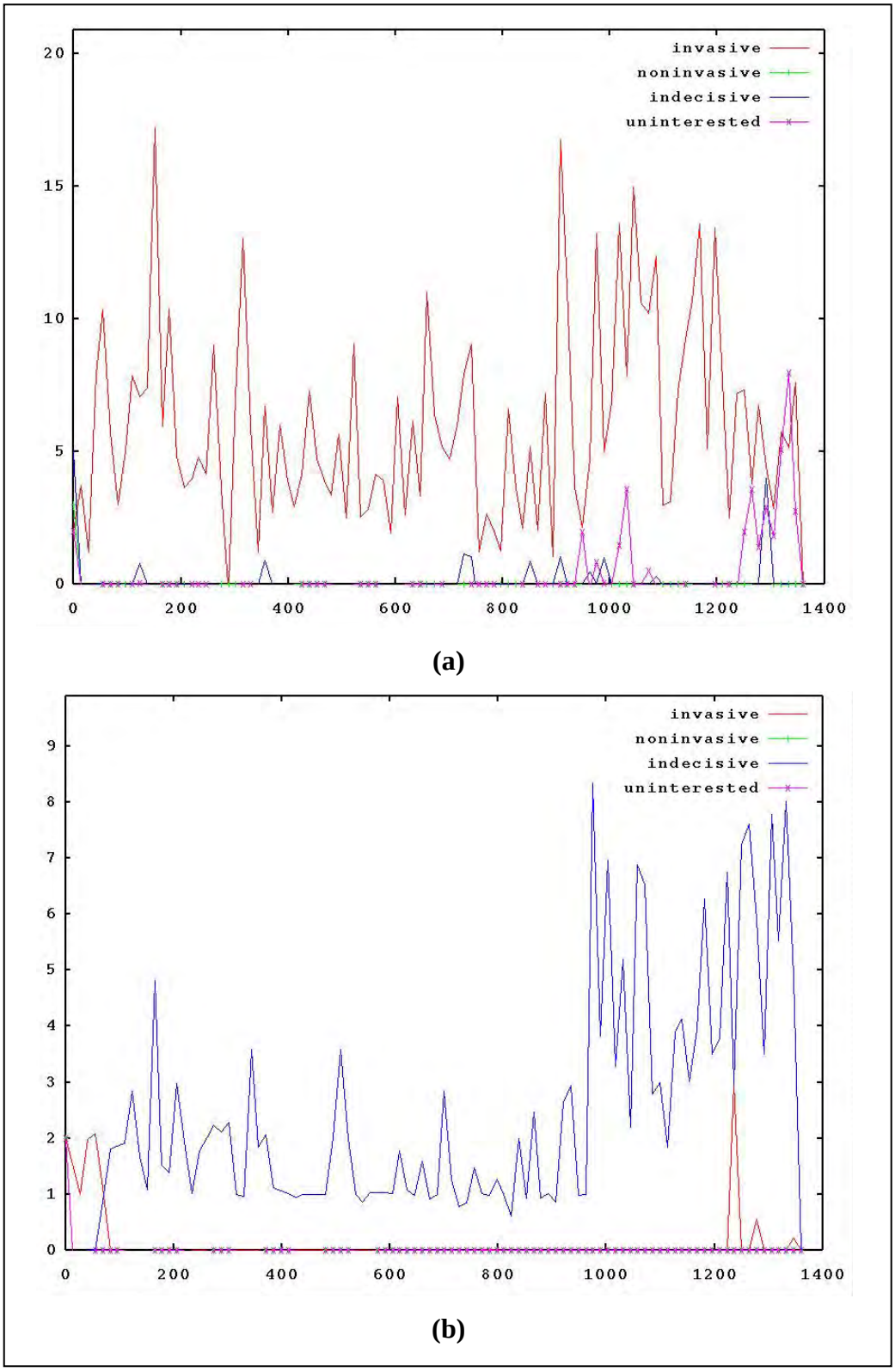


Figure A-18. Coevolutionary dynamics of enemy territory attraction/repulsion classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

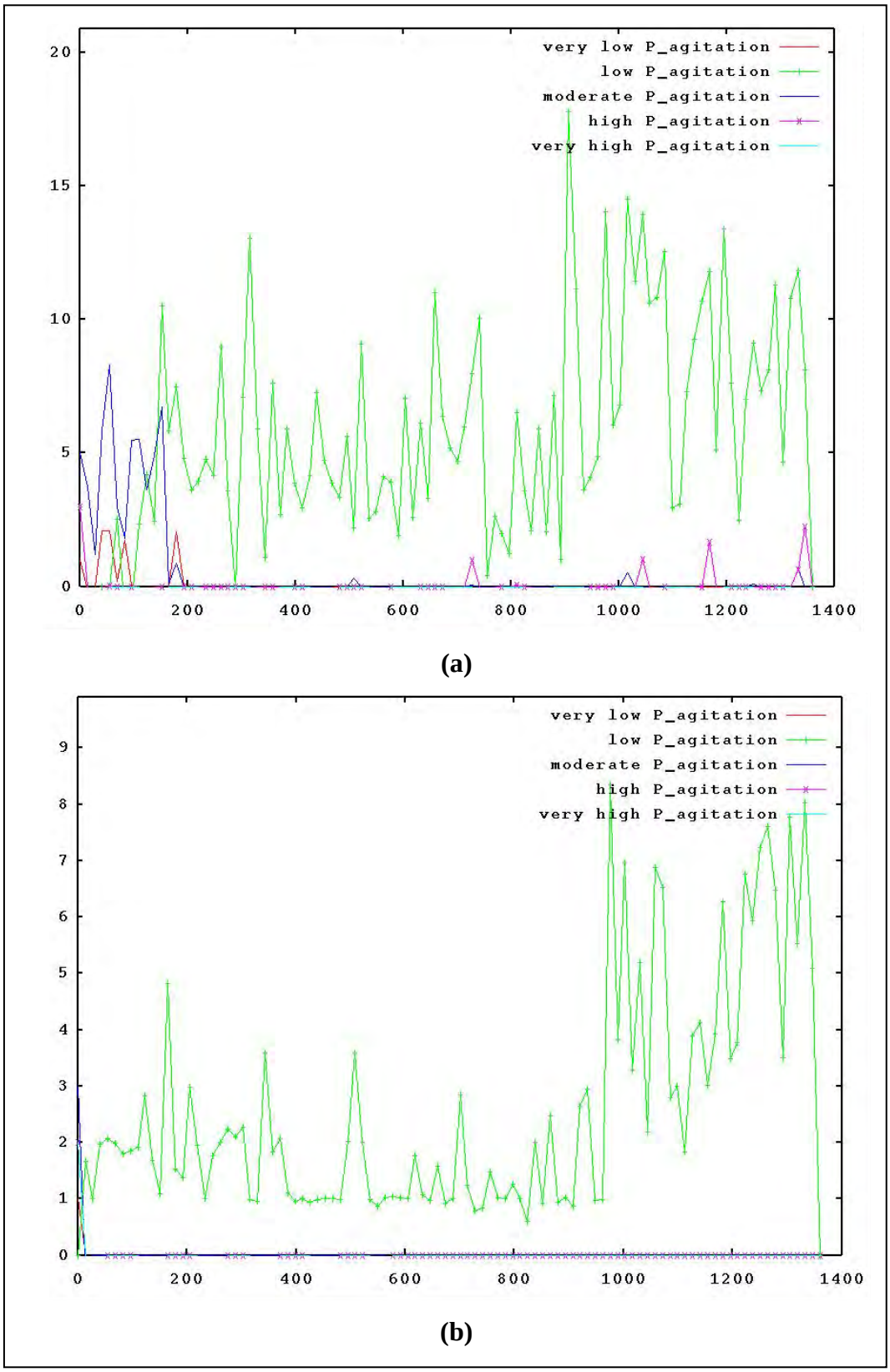


Figure A-19. Coevolutionary dynamics of agitation classes with class population (y-axis) vs. coevolutionary generation (x-axis): (a) Blue unit and (b) Red unit.

Table A-1. Combat agent classes based on gene G_0 (firepower range).

Combat Agent Class	Gene G_0 (Firepower Range)
no firepower	0
very short-range firepower	1–5 cells
short-range firepower	6–10 cells

Table A-2. Combat agent classes based on gene G_1 (single-shot P_{hit}).

Combat Agent Class	Gene G_1 (Single-Shot P_{hit})
very low P_{hit}	$0 \leq G_1 \leq 0.20$
low P_{hit}	$0.20 < G_1 \leq 0.40$
moderate P_{hit}	$0.40 < G_1 \leq 0.60$
high P_{hit}	$0.60 < G_1 \leq 0.80$
very high P_{hit}	$0.80 < G_1 \leq 1.00$

Table A-3. Combat agent classes based on gene G_2 (firepower strength).

Combat Agent Class	Gene G_2 (Firepower Strength)
very low-fire strength	$0 \leq G_2 \leq 4$
low-fire strength	$4 < G_2 \leq 8$
moderate-fire strength	$8 < G_2 \leq 12$
high-fire strength	$12 < G_2 \leq 16$
very high-fire strength	$16 < G_2 \leq 20$

Table A-4. Combat agent classes based on gene G_3 (firepower number of rounds).

Combat Agent Class	Gene G_3 (Firepower Number of Rounds)
very low-fire limit	$0 \leq G_3 \leq 51$
low-fire limit	$51 < G_3 \leq 102$
moderate fire limit	$102 < G_3 \leq 153$
high-fire limit	$153 < G_3 \leq 204$
very high-fire limit	$204 < G_3 \leq 255$

Table A-5. Combat agent classes based on gene G_4 (armor strength).

Combat Agent Class	Gene G_4 (Armor Strength)
very low-armor strength	$0 \leq G_4 \leq 4$
low-armor strength	$4 < G_4 \leq 8$
moderate-armor strength	$8 < G_4 \leq 12$
high-armor strength	$12 < G_4 \leq 16$
very high-armor strength	$16 < G_4 \leq 20$

Table A-6. Combat agent classes based on gene G_5 (sensor range).

Combat Agent Class	Gene G_5 (Sensor Range)
very short-range sensor	$0 \leq G_5 \leq 8$ cells
short-range sensor	8 cells $< G_5 \leq 16$ cells
medium-range sensor	16 cells $< G_5 \leq 24$ cells
long-range sensor	24 cells $< G_5 \leq 32$ cells
very long-range sensor	32 cells $< G_5 \leq 40$ cells

Table A-7. Combat agent classes based on gene G_{11} (carried land mines).

Combat Agent Class	Gene G_{11} (Carried Land Mines)
very few land mines	$0 \leq G_{11} \leq 51$
few land mines	$51 < G_{11} \leq 102$
moderate number of land mines	$102 < G_{11} \leq 153$
many land mines	$153 < G_{11} \leq 204$
very many land mines	$204 < G_{11} \leq 255$

Table A-8. Combat agent classes based on gene G_{12} (carried RF jammer bombs).

Combat Agent Class	Gene G_{12} (Carried RF Jammer Bombs)
very few jammer bombs	$0 \leq G_{12} \leq 51$
few jammer bombs	$51 < G_{12} \leq 102$
moderate number of jammer bombs	$102 < G_{12} \leq 153$
many jammer bombs	$153 < G_{12} \leq 204$
very many jammer bombs	$204 < G_{12} \leq 255$

Table A-9. Combat agent classes based on gene G_{14} (probability of mine/jammer bomb deployment in cell).

Combat Agent Class	Gene G_{14} (Probability Mine/Jammer Bomb Deployment)
very low P_{deploy}	$0 \leq G_{14} \leq 0.10$
low P_{deploy}	$0.10 < G_{14} \leq 0.40$
moderate P_{deploy}	$0.40 < G_{14} \leq 0.60$
high P_{deploy}	$0.60 < G_{14} \leq 0.90$
very high P_{deploy}	$0.90 < G_{14} \leq 1.00$

Table A-10. Combat agent classes based on gene G_{15} (signature).

Combat Agent Class	Gene G_{15} (Signature)
very low visibility	$0 \leq G_{15} \leq 0.10$
low visibility	$0.10 < G_{15} \leq 0.40$
moderate visibility	$0.40 < G_{15} \leq 0.60$
high visibility	$0.60 < G_{15} \leq 0.90$
very high visibility	$0.90 < G_{15} \leq 1.00$

Table A-11. Combat agent classes based on gene G_{16} (number of human crews).

Combat Agent Class	Gene G_{16} (Number of Crews)
no crew	0
one crewmember	1
two crewmembers	2
three crewmembers	3
more than three crewmembers	> 3

Table A-12. Combat agent classes based on gene G_{17} (maximum speed).

Combat Agent Class	Gene G_{17} (Maximum Speed)
immobile	0
very slow	1 – 2
slow	3 – 4
moderate	5 – 6
fast	7 – 8
very fast	9 – 10

Table A-13. Combat agent classes based on gene G_{18} (mode of transportation).

Combat Agent Class	Gene G_{18} (Mode of Transportation)
ground vehicle	0 (signifying a ground-based agent)
air vehicle	1 (signifying an air-based agent)

Table A-14. Combat agent classes based on gene G_{20} (probability of initiating an attack).

Combat Agent Class	Gene G_{20} (Probability Initiating Attack)
very low $P_{initiate\ attack}$	$0 \leq G_{20} \leq 0.10$
low $P_{initiate\ attack}$	$0.10 < G_{20} \leq 0.40$
moderate $P_{initiate\ attack}$	$0.40 < G_{20} \leq 0.60$
high $P_{initiate\ attack}$	$0.60 < G_{20} \leq 0.90$
very high $P_{initiate\ attack}$	$0.90 < G_{20} \leq 1.00$

Table A-15. Combat agent classes based on a combination of genes G_{21} (friendly attraction) and G_{22} (friendly repulsion).

Combat Agent Class	Genes G_{21} and G_{22} (Friendly Attraction/Repulsion)
collaborative	$G_{21} > 0.50$ and $G_{22} \leq 0.50$
noncollaborative	$G_{21} \leq 0.50$ and $G_{22} > 0.50$
indecisive	$G_{21} > 0.50$ and $G_{22} > 0.50$
uninterested	$G_{21} \leq 0.50$ and $G_{22} \leq 0.50$

Table A-16. Combat agent classes based on a combination of genes G_{23} (enemy attraction) and G_{24} (enemy repulsion).

Combat Agent Class	Genes G_{23} and G_{24} (Enemy Attraction/Repulsion)
noncovert	$G_{23} > 0.50$ and $G_{24} \leq 0.50$
covert	$G_{23} \leq 0.50$ and $G_{24} > 0.50$
indecisive	$G_{23} > 0.50$ and $G_{24} > 0.50$
unconcerned	$G_{23} \leq 0.50$ and $G_{24} \leq 0.50$

Table A-17. Combat agent classes based on a combination of genes G_{25} (friendly territory attraction) and G_{26} (friendly territory repulsion).

Combat Agent Class	Genes G_{25} and G_{26} (Friendly Territory Attraction/Repulsion)
defensive	$G_{25} > 0.50$ and $G_{26} \leq 0.50$
nondefensive	$G_{25} \leq 0.50$ and $G_{26} > 0.50$
indecisive	$G_{25} > 0.50$ and $G_{26} > 0.50$
unconcerned	$G_{25} \leq 0.50$ and $G_{26} \leq 0.50$

Table A-18. Combat agent classes based on a combination of genes G_{27} (enemy territory attraction) and G_{28} (enemy territory repulsion).

Combat Agent Class	Genes G_{27} and G_{28} (Enemy Territory Attraction/Repulsion)
invasive	$G_{27} > 0.50$ and $G_{28} \leq 0.50$
noninvasive	$G_{27} \leq 0.50$ and $G_{28} > 0.50$
indecisive	$G_{27} > 0.50$ and $G_{28} > 0.50$
uninterested	$G_{27} \leq 0.50$ and $G_{28} \leq 0.50$

Table A-19. Combat agent classes based on gene G_{29} (probability of agitation).

Combat Agent Class	Gene G_{29} (Probability of Agitation)
very low $P_{agitation}$	$0 \leq G_{29} \leq 0.10$
low $P_{agitation}$	$0.10 < G_{29} \leq 0.40$
moderate $P_{agitation}$	$0.40 < G_{29} \leq 0.60$
high $P_{agitation}$	$0.60 < G_{29} \leq 0.90$
very high $P_{agitation}$	$0.90 < G_{29} \leq 1.00$

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
February 2003	Final	October 2000–December 2001

4. TITLE AND SUBTITLE
Agent-Based Modeling of a Network-Centric Battle Team Operating Within an Information Operations Environment

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Brian G. Ruth and J. Dana Eckart*

5d. PROJECT NUMBER
622618H80

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-SL-EA
Aberdeen Proving Ground, MD 21005-5068

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-2913

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
*Virginia Bioinformatics Institute
1880 Pratt Drive, Bldg. XV
Blacksburg, VA 24061

14. ABSTRACT
A model developed to analyze the emergent behavior of a network-centric battle team undergoing hostile information operations (IO) stress events is presented. Networked battlefield platforms are modeled as mobile semiautonomous agents that operate within a cellular automata (CA) lattice. The CA form a discrete spatially extended dynamical system consisting of a parallel networked lattice of computational cells in two dimensions. A software framework that combines CA-based agents with a genetic algorithm was developed in order to explore the dynamics of two opposing but "coevolving" units of networked combat agents. Simulation results using two variants of the CA-based combat agent model, both of which include IO stress in the form of radio frequency communications jamming, are analyzed and discussed.

15. SUBJECT TERMS
cellular automata, agent-based model, genetic algorithm, network-centric warfare, information operations

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Brian Ruth
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL	130	19b. TELEPHONE NUMBER (*Include area code*) (410) 278-3782

NO. OF COPIES	ORGANIZATION
2	DEFENSE TECHNICAL INFORMATION CENTER DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218
1	COMMANDING GENERAL US ARMY MATERIEL CMD AMCRDA TF 5001 EISENHOWER AVE ALEXANDRIA VA 22333-0001
1	INST FOR ADVNCD TCHNLGY THE UNIV OF TEXAS AT AUSTIN 3925 W BRAKER LN STE 400 AUSTIN TX 78759-5316
1	US MILITARY ACADEMY MATH SCI CTR EXCELLENCE MADN MATH THAYER HALL WEST POINT NY 10996-1786
1	DIRECTOR US ARMY RESEARCH LAB AMSRL D DR D SMITH 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RESEARCH LAB AMSRL CS IS R 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRL CI OK TL 2800 POWDER MILL RD ADELPHI MD 20783-1197
3	DIRECTOR US ARMY RESEARCH LAB AMSRL CS IS T 2800 POWDER MILL RD ADELPHI MD 20783-1197

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
2	DIR USARL AMSRL CI LP (BLDG 305)

NO. OF COPIES ORGANIZATION

1 OASD C3I
J BUCHHEISTER
RM 3D174
6000 DEFENSE PENTAGON
WASHINGTON DC 20301-6000

1 OUSD(AT)/S&T AIR WARFARE
R MUTZELBURG
RM 3E139
3090 DEFENSE PENTAGON
WASHINGTON DC 20301-3090

1 OUSD(AT)/S&T LAND WARFARE
A VIILU
RM 3B1060
3090 DEFENSE PENTAGON
WASHINGTON DC 20310-3090

1 UNDER SECY OF THE ARMY
DUSA OR
ROOM 2E660
102 ARMY PENTAGON
WASHINGTON DC 20310-0102

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECHLGY
SAAL ZP ROOM 2E661
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 ASST SECY ARMY
ACQSTN LOGISTICS & TECHLGY
SAAL ZS ROOM 3E448
103 ARMY PENTAGON
WASHINGTON DC 20310-0103

1 DIRECTOR FORCE DEV
DAPR FDZ
ROOM 3A522
460 ARMY PENTAGON
WASHINGTON DC 20310-0460

1 US ARMY TRADOC ANL CTR
ATRC W
A KEINTZ
WSMR NM 88002-5502

1 USARL
AMSRL SL M
J PALOMO
WSMR NM 88002-5513

NO. OF COPIES ORGANIZATION

1 USARL
AMSRL SL EA
R FLORES
WSMR NM 88002-5513

1 USARL
AMSRL SL EI
J NOWAK
FT MONMOUTH NJ 07703-5601

ABERDEEN PROVING GROUND

1 US ARMY DEV TEST COM
CSTE DTC TT T
APG MD 21005-5055

1 US ARMY EVALUATION CENTER
CSTE AEC SVE
R BOWEN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CENTER
CSTE AEC SVE S
R POLIMADEI
4120 SUSQUEHANNA AVE
APG MD 21005-3013

1 US ARMY EVALUATION CENTER
CSTE AEC SV L
R LAUGHMAN
4120 SUSQUEHANNA AVE
APG MD 21005-3013

12 DIR USARL
AMSRL SL
DR WADE
J BEILFUSS
AMSRL SL B
J FRANZ
P TANENBAUM
L WILSON
AMSRL SL BB
D FARENWALD
M RITONDO
AMSRL SL BD
J MORRISSEY

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND (CONT)

AMSRL SL BE
D BELY
AMSRL SL E
M STARKS
AMSRL SL EC
J FEENEY
E PANUSKA

NO. OF COPIES	ORGANIZATION
1	US ARMY TRADOC BATTLE LAB INTEGRATION TECH & CONCEPTS DIR ATCD B FT MONROE VA 23651-5850
1	US ARMY ARMAMENT RDEC AMSTA AR TD M FISETTE BLDG 1 PICATINNY ARSENAL NJ 07806-5000
1	NATICK SOLDIER CENTER SBCN T P BRANDLER KANSAS STREET NATICK MA 01760-5056
1	US ARMY MISSILE RDEC AMSAM RD MG RSA AL 35898-5420
1	US ARMY TANK AUTOMTV RDEC AMSTA TR J CHAPIN WARREN MI 48397-5000
1	US ARMY INFO SYS ENGRG CMD AMSEL IE TD DR F JENIA FT HUACHUCA AZ 85613-5300
1	US ARMY SIM TRNG INST CMD AMSTI CG M MACEDONIA 12350 RSCH PARKWAY ORLANDO FL 32826-3726
7	INFORMATION SCIENCES TEAM PHYSICAL SCIENCE LAB DR R BERNSTEIN JR DR R SMITH DR M COOMBS DR D HOSKINS DR A POGEL DR S SCHMIDT J ROBEY PO BOX 30002 LAS CRUCES NM 88003-8002

NO. OF COPIES	ORGANIZATION
1	BOOZ ALLEN & HAMILTON INC P MURPHY 4001 FAIRFAX DRIVE SUITE 750 ARLINGTON VA 22203
1	DIRECTOR US ARMY RESEARCH OFFICE 4300 S MIAMI BLVD RTP NC 27709
1	VIRGINIA BIOINFORMATICS INST VIRGINIA TECH DR J D ECKART 1880 PRATT DR BLDG XV (0477) BLACKSBURG VA 24061
4	US ARMY RESEARCH LAB AMSRL SL EA DR K MORRISON DR P DJANG D LANDIN T READER WSMR NM 88002-5513
1	US ARMY RESEARCH LAB AMSRL SL EI J LURSKI FT MONMOUTH NJ 07703-5601
3	US ARMY RESEARCH LAB AMSRL CI AP 2800 POWDER MILL RD ADELPHI MD 20783-1197

ABERDEEN PROVING GROUND

1	SBCCOM RDEC AMSSB RTD J ZARZYCKI 5183 BLACKHAWK RD APG MD 21010-5424

NO. OF
COPIES ORGANIZATION

ABERDEEN PROVING GROUND (CONT)

18 DIR USARL
AMSRL SL B
S JUARASCIO
B WARD
R SANDMEYER
AMSRL SL BE
J ANDERSON
R BOWERS
E FIORAVANTE
AMSRL SL EA
D BAYLOR
R ZUM BRUNNEN
B RUTH (10 CPS)